Trabajo Fin de Grado

Aplicación de realidad aumentada para la monitorización y control de un equipo de electrónica de potencia

Augmented reality application for the monitoring and control of power electronic equipment

Autor

Gonzalo Berné Melguizo

Director

Juan Almajano Francoy

Ponente

Rubén Béjar Hernández

Grado en Ingeniería Informática
Escuela de Ingeniería y Arquitectura
2021

Resumen

La realidad aumentada (RA) es el término que se utiliza para describir la experiencia interactiva donde el entorno del mundo real se mejora con información perceptiva generada por computadora en forma de objetos virtuales.

Se lleva trabajando varios años con la realidad aumentada por lo que no es una tecnología novedosa. Sin embargo, sí que está considerada una tecnología emergente ya que todavía no se ha alcanzado su máximo potencial. Hoy en día, las empresas siguen buscando y analizando ámbitos donde poder aplicar el uso de esta herramienta, para otorgar a los usuarios nuevas formas de interacción en diferentes escenarios.

En el caso expuesto en este trabajo, las empresas Endesa y Fundación CIRCE requerían una aplicación para dispositivos móviles y tablets (y en un futuro gafas con RA) que, utilizando la realidad aumentada, monitorizara y controlara un equipo de electrónica de potencia. La idea que se tiene con esta tecnología es otorgar la posibilidad de ayudar a operarios a realizar labores de mantenimiento de una forma rápida y segura fusionando el entorno real con el virtual. La aplicación desarrollada en este trabajo es un primer acercamiento al uso de esta tecnología en este contexto.

Se ha utilizado el *framework* Unity y el kit de desarrollo de software Vuforia para construir una aplicación que funcione en dispositivos con sistema operativo Android.

Índice

Índice de figuras

1. Introducción

El presente documento tiene como objetivo presentar toda la información relacionada con el Trabajo de Fin de Grado (TFG) titulado Aplicación de realidad aumentada para la monitorización y control de un equipo de electrónica de potencia.

Este trabajo pertenece a la titulación de Grado en Ingeniería Informática de la Universidad de Zaragoza, ha sido realizado por el alumno Gonzalo Berné Melguizo, dirigido y supervisado por Juan Almajano Francoy, perteneciente a Fundación CIRCE, y actúa como ponente Rubén Béjar Hernández, del Departamento de Informática e Ingeniería de Sistemas.

A continuación se va a explicar el contexto en el que se ha realizado este trabajo junto con un esquema de la estructura del documento.

1.1. El problema

La electrónica de potencia es la disciplina tecnológica que trata de la conversión eficiente de energía eléctrica, haciendo uso del control que permiten los equipos electrónicos, a la forma en que la necesitan los usuarios para su consumo.

Muchos productos de alta potencia utilizan la electrónica de potencia, dado que ha alcanzado un lugar importante en la tecnología actual. En el caso de este proyecto, se toma como referencia un equipo de electrónica de potencia, financiado por Endesa y desarrollado por Fundación CIRCE, encargado de balancear las cargas de la línea de distribución para estabilizar y equilibrar el consumo eléctrico en las distintas fases. Este equipo está situado en la ciudad de Málaga. El equipo cuenta con una placa hardware con diversos componentes instalados y un Linux embebido que se encarga de la monitorización y del control del equipo de electrónica de potencia.

El objetivo de este TFG es crear una aplicación que sea capaz de monitorizar y actuar sobre el dispositivo mediante una interfaz de realidad aumentada a través de un teléfono móvil o tablet.

La realidad aumentada [1] (RA) es el término que se usa para describir al conjunto de tecnologías que permiten que un usuario visualice parte del mundo real a través de un dispositivo tecnológico con información gráfica añadida por este. El dispositivo añade información virtual a la información física ya existente, es decir, una parte virtual aparece en la realidad a través de una pantalla. De esta manera los elementos físicos tangibles se combinan con elementos virtuales, creando así una realidad aumentada en tiempo real.

Ya se lleva varios años hablando de este concepto y sus aplicaciones, pero todavía no está implementada tan extensamente como podríamos pensar, es decir, sigue siendo una tecnología emergente. Es por ello por lo que empresas como Endesa y CIRCE están empezando a lanzar proyectos utilizando esta tecnología ya que puede ser muy interesante el alcance al que puede llegar. Esto se debe a que cuentan con productos y equipos propios en los que su integración

tiene sentido y puede ser beneficiosa, y porque hoy en día hay varias herramientas y frameworks que facilitan el desarrollo.

La idea de estas empresas es que, en un futuro, el entorno virtual que proporciona la realidad aumentada ayude a los operarios a realizar sus tareas con mayor rapidez, sobre todo si esas tareas son críticas y se necesita conocer en tiempo real el estado de los equipos.

1.2. Estructura del documento

El documento cuenta con una primera parte donde se explica de una forma global y completa el trabajo realizado en este TFG y que incluye este apartado de Introducción. La segunda parte consta de anexos que complementan la información de la memoria.

La sección introductoria sitúa el problema en contexto y las herramientas a utilizar para resolver el mismo. En la sección de análisis del problema se estudia el problema en profundidad analizando el equipo que hay que monitorizar y presentando los requisitos que debe incluir la aplicación. Después, en la sección 3, *Diseño de la solución*, se presenta el diseño ideado para la aplicación incluyendo la arquitectura del sistema con los datos y objetos que se utilizan y el diseño final de la interfaz de usuario. La sección de *Implementación* habla de todos los aspectos y decisiones llevadas a cabo a la hora de implementar la aplicación siguiendo el diseño. En la sección de *Gestión del proyecto* se expone la planificación y división de las tareas realizadas. Y, para terminar, en la sección de *Conclusiones y trabajo futuro*, se expone el alcance que podría tener la realidad aumentada en este campo y unas conclusiones que valoran el trabajo efectuado y el aprendizaje realizado.

2. Análisis del problema

2.1. Equipo al que se conecta la aplicación

Endesa es una empresa española proveedora de energía eléctrica, que se encarga de construir redes eléctricas por todo el país para poder suministrar energía a todo tipo de instalaciones. Dicha energía es transferida mediante líneas de distribución, que están formadas por fases. Cuando las subestaciones eléctricas que generan la energía tienen que distribuirla a diferentes puntos, es probable que el consumo de cada fase sea desequilibrado, lo que puede provocar una sobrecarga en las líneas, caídas de tensión desiguales y que la red de distribución pueda sufrir un desbalanceo. Para ello se intenta buscar soluciones mediante el desarrollo de equipos que puedan garantizar un consumo equilibrado.

En este caso concreto, se ha actuado sobre una línea de distribución trifásica que suministra energía eléctrica en un barrio de la ciudad de Málaga (Foto de la caja donde está el equipo en la Figura 1), ya que el consumo de las tres fases en la red no era estable. Para ello, mediante un proyecto financiado por Endesa, Fundación CIRCE ha diseñado y desarrollado un equipo que mediante el control en electrónica de potencia es capaz de balancear las cargas de la línea de distribución para conseguir una estabilización del consumo en la red.

Este equipo calcula las cargas que debe balancear y actúa sobre las fases gracias a dos dispositivos (analizadores de red) que están situados sobre la línea de distribución. El primero de los dispositivos se denomina “Analyzer” y es el encargado de leer los datos antes de que el equipo equilibre las fases. El segundo dispositivo se llama “Grid” y actúa sobre la línea según las consignas recibidas por el equipo devolviéndole los valores finales de las fases una vez balanceadas las cargas.

Figura 1: Foto de la ubicación del equipo instalado en Málaga

En el Anexo A se encuentra más información al respecto del funcionamiento de este equipo.

2.2. Requisitos

Las tareas a realizar en esta aplicación se han dividido en requisitos. La captura de estos requisitos se ha llevado a cabo siguiendo una aproximación de algunas de las técnicas consideradas dentro del marco de las metodologías ágiles. Al comienzo del proyecto se definieron los requisitos más importantes que forman la base de la aplicación. A partir del seguimiento y de las reuniones realizadas por las empresas interesadas se daba *feedback* para ir ampliando y adaptando el listado de requisitos a las necesidades reales del proyecto sin desviarse de la idea principal.

El listado de requisitos no ha sido un listado cerrado desde el principio, pero, aunque se han ido ampliando y adaptando según iba avanzando el desarrollo, la idea principal siempre ha sido la misma. Por ejemplo, el concepto de modos de visualización y su integración en la aplicación no se hizo al principio, se decidió más adelante en el proceso del desarrollo según se evolucionaba con el diseño de la GUI.

2.2.1. Glosario

En esta sección se definen algunos términos que se usan en los requisitos.

- **Equipo**: Equipo de electrónica de potencia (EnergyBox) al que se conecta la aplicación.
- **Línea**: Fases de la distribución de energía eléctrica, que el equipo se encarga de balancear y estabilizar.
- **Objeto**: Componente virtual que se visualiza a través de la aplicación mezclándose con el entorno real. A través de la aplicación se puede interactuar con algunos de los objetos.
- **Baliza**: Marca que la aplicación puede identificar a través de la cámara del dispositivo para poder situar los objetos virtuales en el espacio tras reconocer el entorno en el que se sitúa.
- **Modo de visualización**: Distintos menús que cambian la forma en que la se muestra la información del equipo al usuario. Ejemplos: “Monitorización del equipo”, “Gráficos”.
- **Flotante**: Objeto virtual que la aplicación sitúa en el entorno real a través de la pantalla, dándole una forma de tres dimensiones.
- **Rígido**: Objeto que se muestra sobre la pantalla del dispositivo en dos dimensiones.

2.2.2. Requisitos funcionales

A continuación se exponen los requisitos que se han definido para construir una aplicación de realidad aumentada para móviles y tablets Android. El sistema tiene que ser capaz de monitorizar un equipo de control de electrónica de potencia mostrando al usuario objetos 3D en el espacio virtual que representen al equipo o que aporten información extra como gráficos.

Sistema	
Sistema_RF1	El sistema permitirá la detección de una baliza para situar el equipo en el espacio que visualiza la cámara del dispositivo.
Sistema_RF2	El sistema permitirá cambiar entre varios modos de visualización a través de un menú flotante. Siempre un modo a la vez.
Sistema_RF3	El sistema constará con un menú rígido que permitirá cambiar entre los modos de visualización. Siempre un modo a la vez.

Modo de visualización: Monitorización del equipo	
Monitorización_RF1	El sistema permitirá visualizar el equipo como un objeto en el espacio.
Monitorización_RF2	El sistema permitirá visualizar el estado del equipo y de sus líneas en tiempo real.
Monitorización_RF3	El sistema permitirá al usuario cambiar el estado del equipo entre “Funcionando”, “Pausa” y “Reinicio”.
Monitorización_RF4	Si el equipo se encuentra en “Error”, el sistema deberá informar de su estado, mostrar el código del error y un identificador textual del error.

Modo de visualización: Gráficos	
Gráficos_RF1	El sistema permitirá visualizar gráficos de barras y gráficos de líneas como objetos en el espacio.
Gráficos_RF2	El sistema permitirá cambiar entre los tipos de gráficos. Solo se podrá visualizar un tipo de gráfico a la vez.
Gráficos_RF3	El sistema permitirá cambiar la magnitud a visualizar en el gráfico de barras entre “Potencia de equipo”, “Potencia Reactiva”, “Potencia Activa”, “Voltaje” y “Corriente”.
Gráficos_RF4	El sistema mostrará en el gráfico de líneas la “Potencia Aparente” entre las tres líneas del equipo y la línea neutro.
Gráficos_RF5	El sistema permitirá visualizar en el gráfico de líneas los valores que alcanzan las líneas antes y después de ser balanceadas por el equipo para así poder comparar como actúa el sistema en ellas.

Panel de información	
Información_RF1	El sistema permitirá visualizar en un panel de información independiente de los modos de visualización distintas medidas en tiempo real. Dichas medidas son: “Potencia de equipo”, “Potencia Reactiva”, “Potencia Activa”, “Voltaje” y “Corriente”.
Información_RF2	El sistema permitirá filtrar la información mostrada en el panel de información entre las tres líneas o entre las variables disponibles (“Potencia de equipo”, “Potencia Reactiva”, “Potencia Activa”, “Voltaje” y “Corriente”).

2.2.3. Requisitos no funcionales

Requisitos no funcionales	
RNF1	La aplicación funcionará en dispositivos Android versión 6 o superior.
RNF2	La aplicación tendrá una interfaz de realidad aumentada que se activará al detectar una imagen sobre una superficie.
RNF3	El estado en el que se encuentra el equipo deberá ser lo primero que se debe visualizar al ejecutar la aplicación.

2.3. Interfaces de usuario

Durante la fase de análisis se propusieron algunas ideas preliminares sobre la interfaz de usuario, que luego se concretaron y mejoraron conforme avanzaba el diseño de la aplicación y se iban generando los distintos prototipos. Hasta la fase de diseño no se decidió que hubiera una distinción entre modos de visualización, lo que ha afectado considerablemente a la distribución de los diferentes objetos en el entorno.

Como se muestra en la
Figura 2, la primera idea de interfaz de usuario fue la de dividir el entorno que hay en frente del usuario en tres espacios con diferentes objetivos. Estos espacios son visualizados a través de la cámara del dispositivo tras realizar un reconocimiento de una baliza o marca.

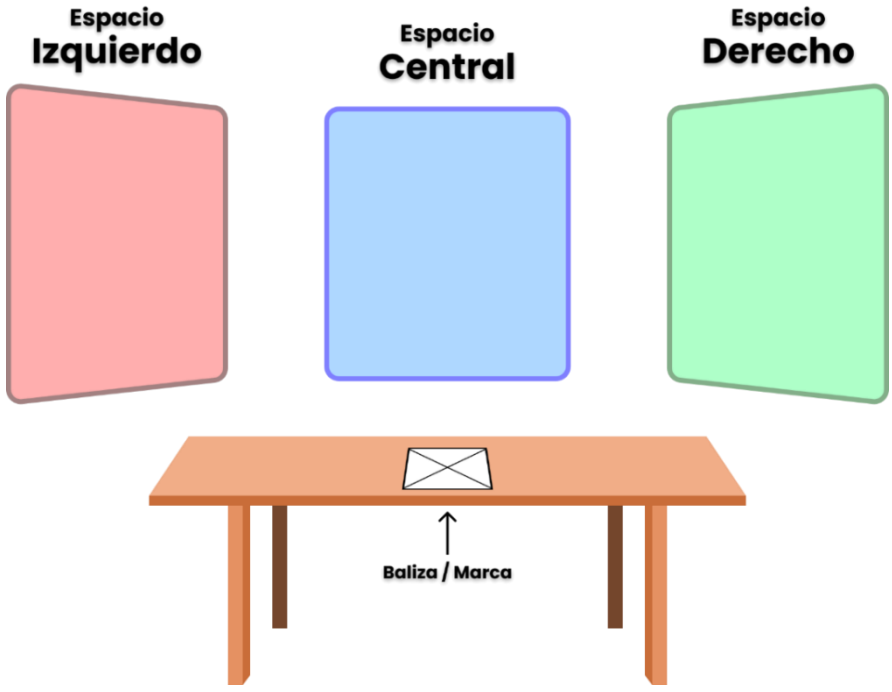


Figura 2: Distribución del espacio virtual en la fase de análisis

Los espacios izquierdo y derecho (imagen de ejemplo en

Figura 3) fueron pensados para la visualización de gráficos.

Figura 3: Ejemplo de gráficos en espacio izquierdo y derecho

El espacio central fue pensado como el lugar donde albergar el panel de actuación general de la aplicación, donde poder ejecutar acciones sobre los gráficos, como rotarlos mediante botones, cambiar el tipo de gráficos a mostrar o sus valores. También se pensó que este espacio central se podría utilizar para mostrar valores más exhaustivos, complementarios, o también como histórico de los datos tratados en los gráficos que se estuvieran mostrando en ese momento en los espacios laterales. En la Figura 4 se muestran imágenes de ejemplo del espacio central.

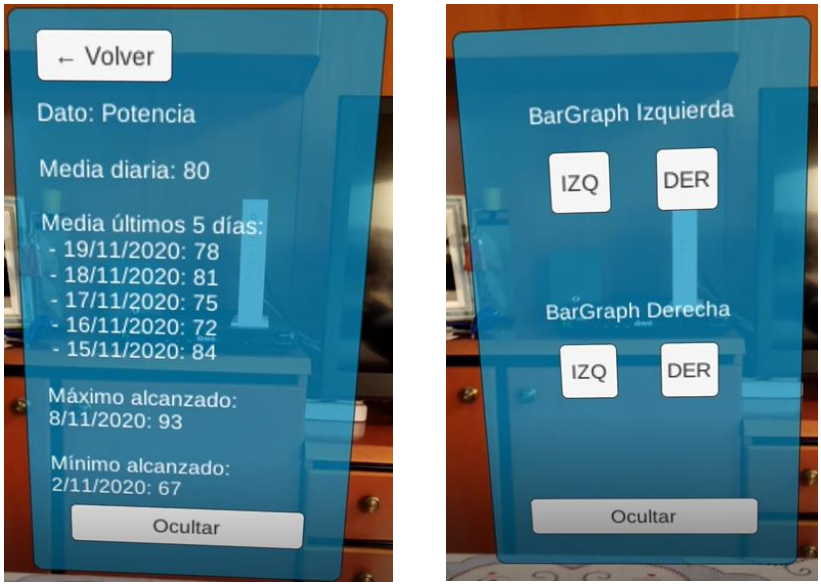


Figura 4: Ejemplo de panel de actuación en espacio central

Aparte de estos espacios ya explicados, en esta etapa del desarrollo ya se pensó en albergar un cuarto espacio donde poder incluir otro tipo de información o de forma diferente. Este otro espacio tiene la particularidad de ser rígido y no flotante como el resto, es decir, aunque la cámara del dispositivo se mueva y enfoque a otro sitio, este espacio siempre se mantiene inmóvil sobre la pantalla. En un principio se pensó en darle un enfoque similar al del espacio central: que sirviera de espacio de actuación respecto a los gráficos de los espacios laterales y de la aplicación en general y que contara también con la posibilidad de mostrar datos extras que complementaran a estos gráficos. La Figura 5 muestra imágenes de ejemplo sobre este espacio rígido.

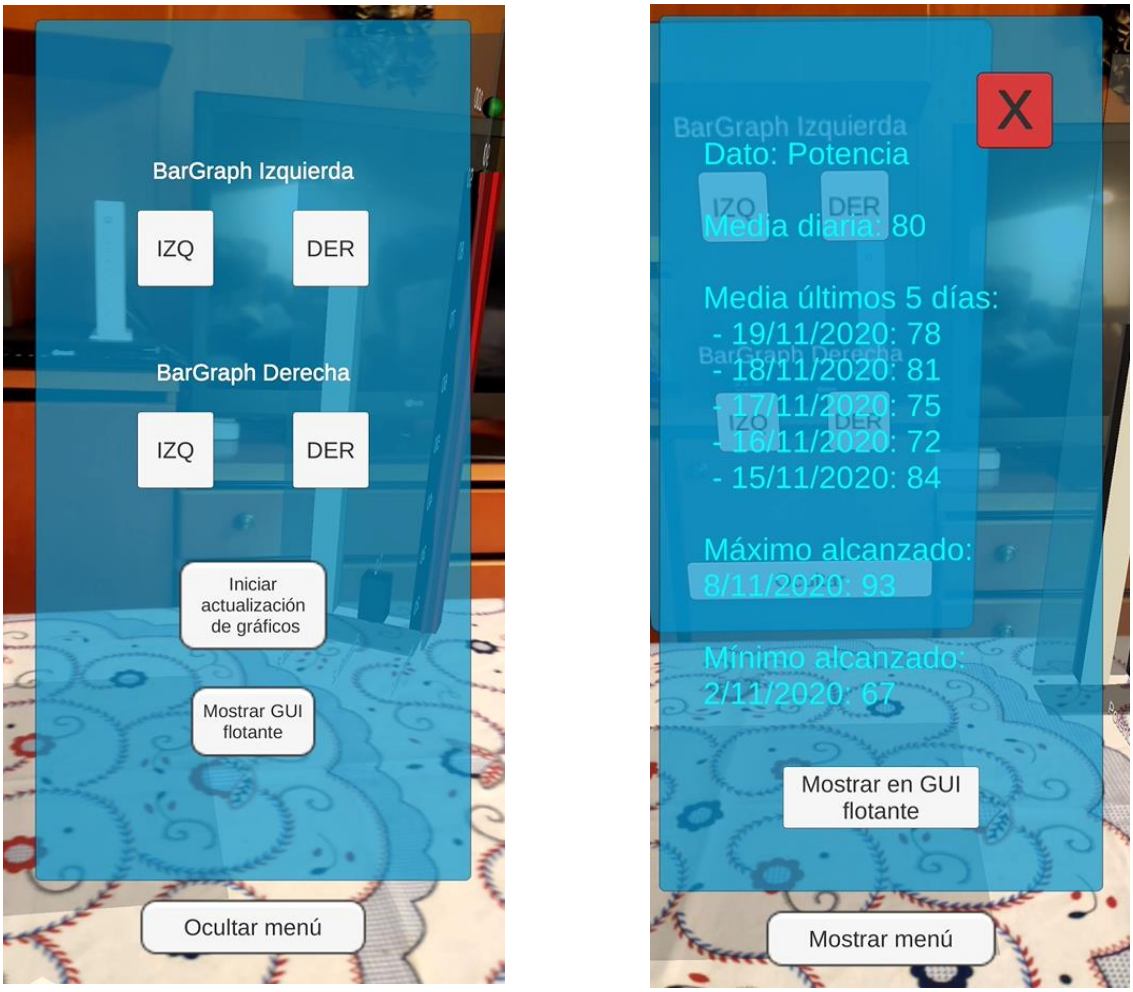


Figura 5: Ejemplo de panel de actuación en espacio rígido

3. Diseño de la solución

3.1. Tecnología

Una vez se definió con las empresas involucradas el alcance del proyecto, se buscaron las tecnologías disponibles en el mercado que encajaran mejor con lo que se requería. Como resultado, las herramientas más importantes utilizadas han sido:

Unity

Unity [2] es uno de los motores de videojuegos y aplicaciones 3D más populares. Cuenta con una licencia gratuita y otra mejorada para estudiantes. Se ha elegido por la capacidad que tiene como sistema de desarrollo y de integración con la realidad aumentada y por su facilidad para exportar un proyecto a sistemas Android (y a otras plataformas). Otros puntos a favor han sido la documentación que aporta y la comunidad que mantiene vivo el framework, ya sea ayudando a otros usuarios o desarrollando nuevas extensiones que están disponibles en una tienda online.

Vuforia

Vuforia [3] es la librería utilizada para hacer funcionar la realidad aumentada en el entorno de desarrollo Unity. Es cierto que hay otras herramientas como ARFoundation [4] que incluye el propio Unity y que se tuvieron en cuenta. Tras compararla con otras se decidió utilizar Vuforia por varias razones: pertenece a la empresa PTC la cual lleva trabajando desde 1985 en innovar en el sector de la industria y la fabricación con nuevas tecnologías, Unity sigue estando enfocado principalmente al desarrollo de videojuegos, por lo que se pensó que Vuforia estaría más preparada para el tipo de aplicación que se requería; la documentación y la información de Vuforia que se encuentra online también está enfocada al desarrollo de aplicaciones y no tanto a videojuegos, además de ser más extensa que otras librerías del mismo ámbito ya que lleva muchos años integrada con Unity; por último, la facilidad que otorga Vuforia para implementar sus funcionalidades en un proyecto de Unity también han sido un factor importante en la decisión.

Visual Studio y C#

Para el desarrollo de los scripts que han otorgado funcionalidad a los objetos de la aplicación se ha utilizado Visual Studio [5] como entorno de programación y C# [6] como lenguaje de programación. Ambas opciones son las que Unity trae por. C# es un lenguaje de programación orientada a objetos cuya sintaxis básica deriva de C y C++ y está desarrollado por Microsoft como parte de la plataforma .NET.

3.2. Arquitectura

Puesto que el proyecto se basa en Unity, para describir su arquitectura es importante conocer cómo se estructura este framework. La próxima sección describe los principales elementos arquitecturales de Unity

3.2.1. Unity

Unity es un motor multiplataforma que ayuda a los desarrolladores a crear videojuegos y aplicaciones 2D y más comúnmente 3D. El framework Unity incluye Unity Editor, el programa donde se editan las escenas, *GameObjects*, componentes y demás artefactos que constituyen la aplicación que se está desarrollando.

Unity, como buen framework, provee un conjunto de herramientas que agilizan el proceso de desarrollo y abstraen al desarrollador de aspectos menos importantes para la idea general del programa que se está construyendo pero que son imprescindibles para el usuario final. Esto es una gran ventaja ya que programar un motor de renderizado, las físicas y la detección de colisiones de objetos u otras partes de la aplicación puede ser realmente complejo.

La existencia de una tienda virtual llamada “Unity Asset Store” otorga muchas opciones a los usuarios. En esta se pueden encontrar diferentes servicios, herramientas o extensiones creadas por el propio equipo de desarrollo de Unity, estudios especializados en el desarrollo de plugins o por los propios usuarios. Esto ayuda a crear aplicaciones mucho más completas e interesantes y a fortalecer la comunidad.

Es importante conocer las partes esenciales de la arquitectura de Unity. La Figura 6 muestra el diagrama de clases de Unity.

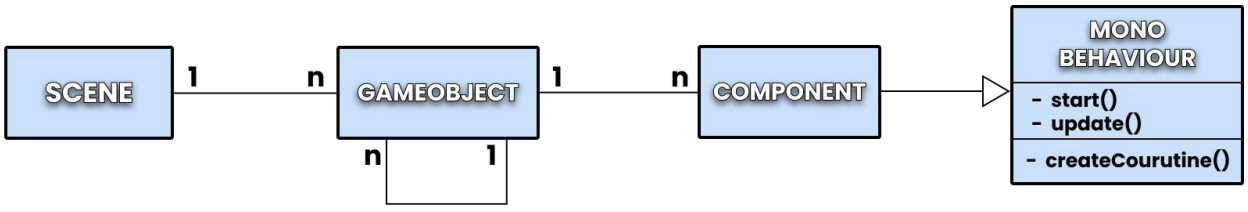


Figura 6: Diagrama de clases de Unity

Una escena (*scene*) es el escenario a partir del cual se sitúan los diferentes elementos de Unity. Los *GameObjects* son los objetos virtuales que aparecen sobre la escena y que el usuario visualiza a través de la pantalla como modelos 3D. Los *GameObjects* se pueden agrupar y crear *GameObjects* más complejos. Los componentes otorgan la funcionalidad y las propiedades a los *GameObjects*, es decir, definen su comportamiento. Estos componentes pueden venir dados por Unity o creados por el usuario a través de *scripts*. Tanto si son de una forma u otra, todos son generados a partir de la clase *Mono Behaviour*.

En el Anexo B se proporciona más información sobre la arquitectura de Unity.

3.2.2. Estilo arquitectónico

Utilizando la infraestructura que proporciona Unity, se ha utilizado el patrón “Modelo - Vista - Presentador” [7] (MVP) para crear el modelo de la arquitectura que sigue la aplicación y que se muestra en el diagrama de la Figura 7.

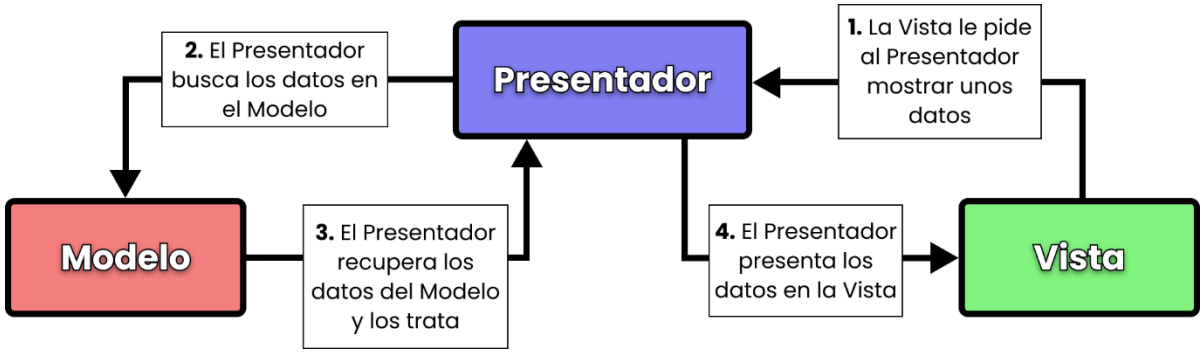


Figura 7: Estilo arquitectónico de la aplicación. Modelo - Vista - Presentador

Como su nombre indica, este estilo arquitectónico expone tres componentes. La Vista exhibe datos y presenta eventos al usuario con los cuales cambiar los datos que se están visualizando. El Presentador actúa sobre el Modelo y la Vista, recupera datos del Modelo y los presenta en la Vista. Por último, el Modelo contiene los datos que pide el Presentador para mostrarlos en la Vista.

En la aplicación desarrollada y siguiendo el modelo de clases de Unity, la Vista son los *GameObjects* que el usuario visualiza a través de la pantalla y que muestran unos datos. El usuario puede interactuar mediante botones incluidos en la Vista para cambiar los datos mostrados, y estas interacciones provocan llamadas en los *scripts* de los *GameObjects* que piden al Modelo los nuevos datos. Estos *scripts* actúan de Presentador. Por último, el Modelo son los datos que la aplicación recupera del equipo de electrónica de potencia.

3.2.3. Modelo de datos

Los datos que utiliza la aplicación son suministrados por el equipo. Siguiendo el patrón MVP presentado en el punto anterior, estos datos son el componente Modelo. Mediante una petición a través de internet (utilizando el protocolo HTTP) a la API que éste tiene expuesta, se reciben unos datos estructurados en formato JSON, basado en texto, que contienen la información en tiempo real del equipo y sus líneas.

Estos datos vienen estructurados en dos vectores o *arrays*. El primero de ellos se corresponde con los valores medidos por el dispositivo *Analyzer* y está formado por pares clave-valor. Estas medidas indican los valores de las líneas antes de que el equipo actúe sobre ellas y hacen posible una comparación del estado de éstas antes y después de pasar por el equipo.

El segundo de los *arrays* también está formado por pares clave-valor, aunque en este caso las medidas son dadas por el dispositivo *Grid*, de tal forma que indican los valores de las líneas después de haber sido balanceadas por el equipo. Los datos del dispositivo *Grid* también incluyen otros valores que no da *Analyzer*, como el código del estado del sistema, la potencia interna que ha usado el equipo en el balanceo de cargas o, en caso de que el equipo estuviera en estado de error, el identificador de este.

A continuación, en la Tabla 1, se detallan los datos recibidos por el dispositivo *Analyzer* que son necesarios para el funcionamiento de la aplicación:

Tabla 1: Datos recibidos del dispositivo Analyzer (antes de pasar por el equipo)

Clave	Valor
React_Pw_1	Potencia Reactiva línea U (Voltiamperios Reactivo, VAr)
React_Pw_2	Potencia Reactiva línea V (Voltiamperios Reactivo, VAr)
React_Pw_3	Potencia Reactiva línea W (Voltiamperios Reactivo, VAr)
Act_Pw_1	Potencia Activa línea U (Vatios, W)
Act_Pw_2	Potencia Activa línea V (Vatios, W)
Act_Pw_3	Potencia Activa línea W (Vatios, W)
V1	Voltaje línea U (Voltios, V)
V2	Voltaje línea V (Voltios, V)
V3	Voltaje línea W (Voltios, V)
A1	Intensidad línea U (Amperios, A)
A2	Intensidad línea V (Amperios, A)
A3	Intensidad línea W (Amperios, A)

La Tabla 2 muestra los datos recibidos por el dispositivo Grid que se utilizan en la aplicación:

Tabla 2: Datos recibidos del dispositivo Grid (después de pasar por el equipo)

Clave	Valor
CPMS_Status_Flag	Código identificador del estado del equipo
CPMS_Major_Error_List_String	Si hay un error, texto que indica el error concreto
Grid_PU_Set	Potencia del equipo línea U (Vatios, W)
Grid_PV_Set	Potencia del equipo línea V (Vatios, W)
Grid_PW_Set	Potencia del equipo línea W (Vatios, W)
PS_QU_Grid	Potencia Reactiva línea U (Voltiamperios Reactivo, VAr)

PS_QV_Grid	Potencia Reactiva línea V (Voltiamperios Reactivo, VAr)
PS_QW_Grid	Potencia Reactiva línea W (Voltiamperios Reactivo, VAr)
PS_PU_Grid	Potencia Activa línea U (Vatios, W)
PS_PV_Grid	Potencia Activa línea V (Vatios, W)
PS_PW_Grid	Potencia Activa línea W (Vatios, W)
PS_VU_Grid	Voltaje línea U (Voltios, V)
PS_VV_Grid	Voltaje línea V (Voltios, V)
PS_VW_Grid	Voltaje línea W (Voltios, V)
PS_IU_Grid	Intensidad línea U (Amperios, A)
PS_IV_Grid	Intensidad línea V (Amperios, A)
PS_IW_Grid	Intensidad línea W (Amperios, A)

La clave CPMS_Status_Flag del dispositivo *Grid* indica el estado del equipo mediante un identificador numérico. En esta aplicación, sólo se representan cuatro estados diferentes del equipo ("Funcionando", "Pausa", "Reiniciando" y "Error") pero algunos abarcan varios códigos identificadores de los que suministra el dispositivo *Grid*, es decir, para el *Grid* hay estados que son diferentes, pero para la aplicación son el mismo.

La Tabla 3 compara los estados de la aplicación con los del dispositivo *Grid*:

Tabla 3: Tabla comparativa de los estados de la aplicación con los datos recibidos por el dispositivo Grid

Estado Aplicación	Estado *Grid*	Código *Grid*
Reiniciando	*Initializing*	1
Pausa	*Stopped*	2
Funcionando	*Preparing*	8
	Balancing	16
	Suplying Reactive Power	32
	Consuming Reactive Power	64
Error	*Minnor Error Stopping*	256
	Minnor Error Stopped	512
	Mayor Error Stopping	1024
	Mayor Error Stopped	2048

3.2.4. Modelo de objetos

En este punto se va a explicar el modelo de objetos que se ha diseñado para desarrollar esta aplicación. Como muestra la Figura 8, este modelo de objetos integra los componentes "Presentador" y "Vista" del patrón MVP.

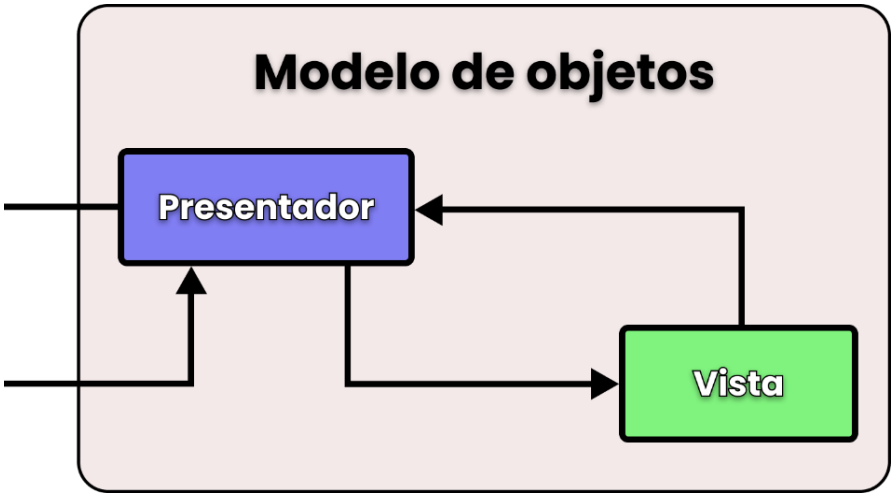


Figura 8: Componentes del patrón MVP que crean el modelo de objetos

Siguiendo los requisitos expuestos, se ha creado un modelo que consta de seis objetos, todos ellos instancias de la clase *GameObject* de Unity. El diagrama de este modelo está disponible en la Figura 9. Estos objetos contienen otros *GameObjects* que forman los modelos 3D (como botones, paneles o los campos de texto). Estos se agrupan en un objeto dependiendo lo que representen y en qué zona estén, para así poder controlar todo el agrupamiento con un solo script. Estos *GameObjects* no solo representan el modelo 3D o cambian su apariencia según lo que les dice el objeto que los contiene, sino que también hay algunos (como los botones) que actúan directamente sobre el funcionamiento del objeto. En resumen, para Unity:

- Los objetos de este modelo son *GameObjects* que hacen de contenedor para el resto de *GameObjects*. Por lo tanto, el modelo 3D que visualiza el usuario de cada uno de estos objetos, es el conjunto de modelos 3D de los *GameObjects* contenidos.
- Los objetos tienen un solo componente: el *script* de control del objeto en particular que controla el comportamiento de los *GameObjects* que contiene.

Un ejemplo es el objeto Graph, que contiene todos los *GameObjects* que crean los gráficos 3D y los paneles para interactuar sobre esos gráficos y, gracias al *script* MainGraph, controla el funcionamiento de todos.

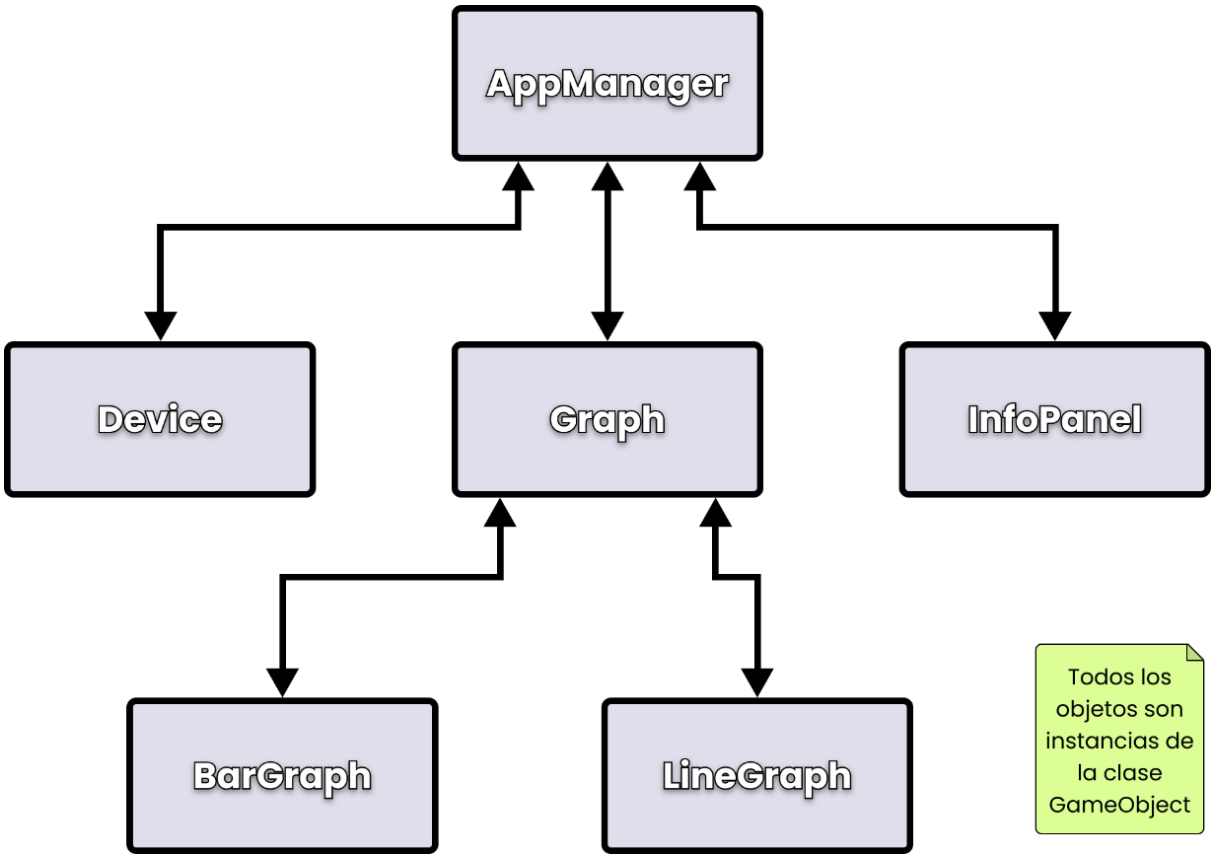


Figura 9: Diagrama de objetos de la aplicación

AppManager

AppManager es el objeto encargado de la comunicación entre la aplicación y la API que expone el equipo.

Su principal función es recuperar los datos disponibles del equipo de control. Esta comunicación se realiza mediante una corrutina que lanza la petición cada 10 segundos, recibe los datos en formato JSON, los transforma, los almacena en una variable, y comunica a los objetos Dispositive, Graph e InfoPanel que hay una actualización de los datos.

Otra comunicación que también realiza este objeto con el equipo es cuando desde la aplicación se quiere actuar directamente con el estado de este.

Device

Device tiene como función mostrar al usuario cuál es el estado del equipo al que está conectado y ofrecer una interfaz para interactuar con el mismo. El script asociado que controla el comportamiento de este objeto se denomina MainDevice.

A este objeto sólo le interesa conocer cuál es el código del estado en el que se encuentra actualmente el equipo de control de potencia, para poder mostrárselo al operario; también recibe del valor la potencia activa con la cual el equipo actúa sobre cada línea. Este valor es el que nos indica el balanceo de cargas que se realiza en el equipo: si es negativo significa que el equipo consume energía de la línea, pero si es positivo significa que está inyectando potencia en esa línea.

La Figura 10 muestra la representación de la Vista de este objeto.

Figura 10: Representación de la Vista del objeto Device

Graph

Graph controla toda la parte del Modo de visualización de gráficos en el sistema. MainGraph es el script que controla este objeto. Es el encargado de recoger los datos, tratarlos y enviárselos a BarGraph o a LineGraph, según sea necesario.

Las magnitudes por las que se puede filtrar en BarGraph son “Potencia de equipo”, “Potencia Reactiva”, “Potencia Activa”, “Voltaje” y “Corriente”. Por parte de LineGraph se muestra la “Potencia Aparente” de las líneas del sistema. Se puede dibujar el gráfico según se quiera visualizar cómo está esta potencia antes del paso de las líneas por el equipo, o después del balanceo de las cargas tras interactuar con el equipo.

- BarGraph:

BarGraph corresponde a un gráfico de barras donde se visualizan diferentes valores que le suministra el objeto Graph.

Para la creación de este objeto se ha utilizado la biblioteca “3D Barchart ViitorCloud”. Esta biblioteca permite la creación de gráficos de barras a partir de unos valores dados. Está preparado para que el gráfico se pueda actualizar dinámicamente, y también se pueden aplicar diversos estilos tanto en las barras 3D como en la secuencia de la actualización del gráfico. Por último, permite la modificación de los scripts de esta biblioteca para que se puedan adaptar a necesidades más concretas.

La Figura 11 muestra la representación de la Vista de este objeto.

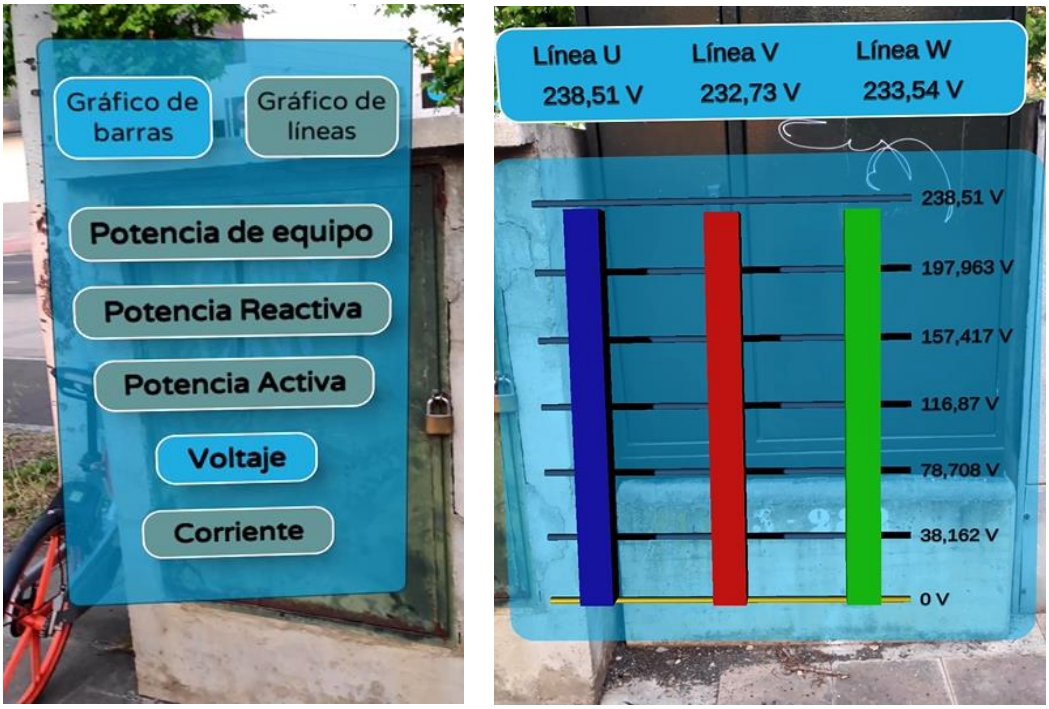


Figura 11: Representación de la Vista del objeto BarGraph

- LineGraph:

LineGraph corresponde al gráfico de líneas que muestra la “Potencia Aparente” de cada una de las líneas del equipo.

Este objeto lo forman dos scripts, GridRenderer (crea el panel que actúa de fondo con las coordenadas sobre donde se apoya el propio gráfico de líneas) y LineRenderer (a partir de los valores dados desde el objeto Graph crea las líneas que representan la “Potencia Aparente” en el sistema).

La Figura 12 muestra la representación de la Vista de este objeto

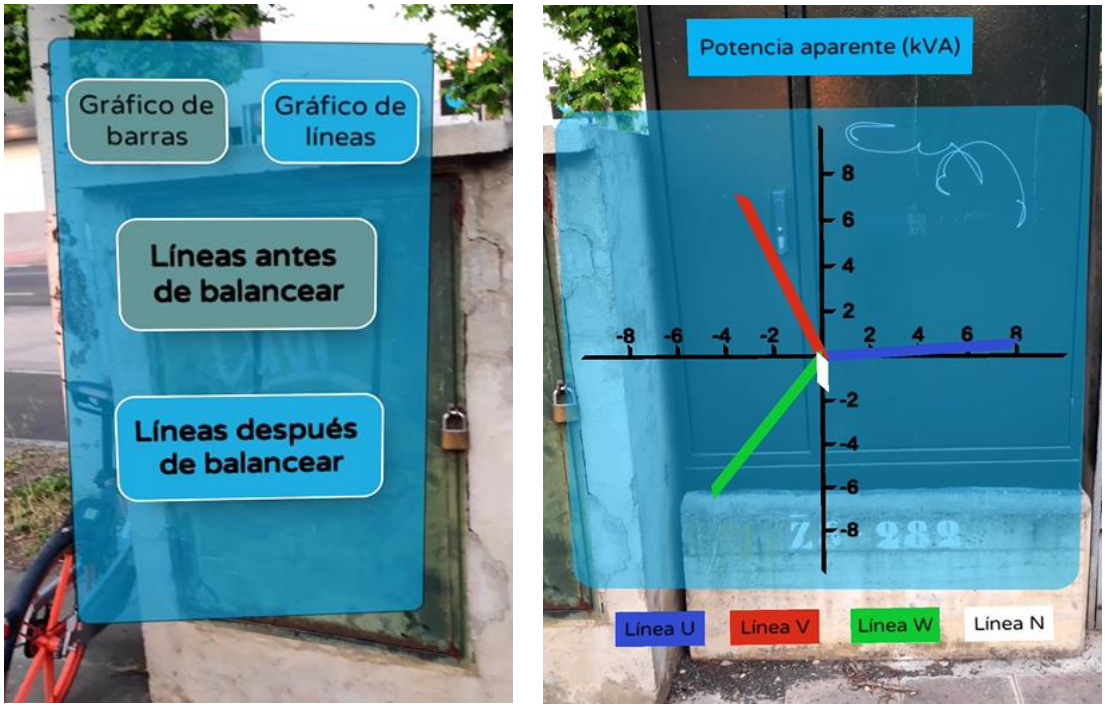


Figura 12: Representación de la Vista del objeto LineGraph

InfoPanel

InfoPanel es el encargado de mostrar los valores de las magnitudes que son más representativas para los usuarios que van a utilizar esta aplicación. Estas magnitudes son “Potencia de equipo”, “Potencia Reactiva”, “Potencia Activa”, “Voltaje” y “Corriente”. A diferencia de los otros objetos, en este los datos simplemente se muestran mediante campos de texto (sin gráficos u otros objetos 3D) agrupados en un panel, cuando InfoPanel recibe una actualización de los datos, realiza su tratamiento y los muestra. El script que controla el comportamiento de este objeto se denomina MainInfo.

La Figura 13 muestra la representación de la Vista de este objeto

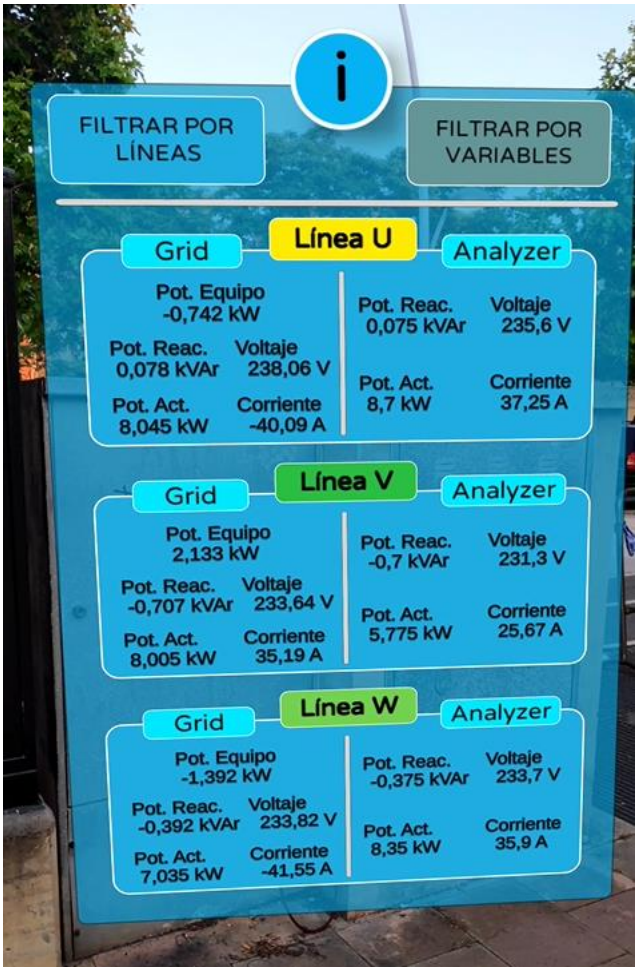


Figura 13: Representación de la Vista del objeto InfoPanel

3.2.5. Comunicación y dinámica del sistema

En este capítulo se va a explicar cómo se realiza la comunicación entre los objetos del sistema presentados en la sección anterior.

Esta comunicación se lleva a cabo a través de los *scripts* asociados a cada objeto. Para que un script encuentre el script de otro objeto debe conocer su identificador, y a partir de ahí puede acceder a las propiedades y métodos disponibles del objeto encontrado.

A continuación se explican las comunicaciones entre los objetos más importantes del sistema.

Actualización de datos en tiempo real

La actualización de datos en tiempo real es la interacción principal del sistema. El objeto AppManager incluye la corrutina que lanza cada 10 segundos una *query* a la API pidiendo una lectura de los datos necesarios. Una vez recibidos los datos en formato JSON, los almacena en una variable y avisa a los objetos Device, Graph e InfoPanel de que hay una nueva actualización de los datos y deberían reflejarla. Cuando son avisados, estos objetos le piden a AppManager la variable con los datos y al recibirlos se actualizan realizando su propio tratamiento de datos.

Durante el desarrollo, se planteó esta otra opción: cuando el objeto AppManager fuera a avisar a los otros tres objetos de que los datos estaban disponibles, se mandaría también la variable con los datos. De esta manera, se evitaba la segunda comunicación donde sólo se solicitaba la variable de los datos. Al final se optó por la otra opción por reutilización de código, ya que hay otros casos donde uno de estos objetos tiene que realizar otro tratamiento de datos y necesita pedir a AppManager la última actualización de los datos (por ejemplo, al pulsar un botón y cambiar el tipo de gráfico o al cambiar la magnitud y los valores a visualizar en el panel de información).

El siguiente diagrama de secuencia de la Figura 14 muestra cómo se realiza la interacción entre los objetos a través de los métodos disponibles en sus scripts al haber una actualización de datos. Se realiza automáticamente una cada 10 segundos.

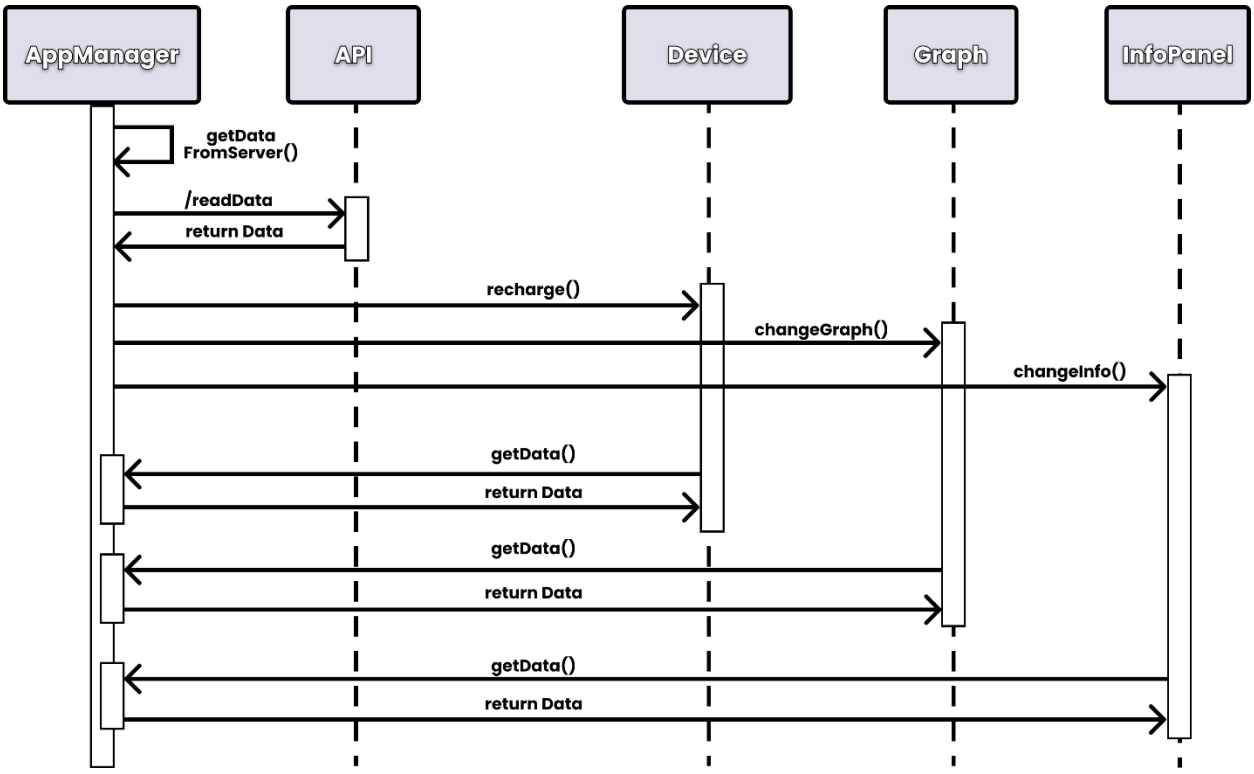


Figura 14: Diagrama de secuencia actualización de datos desde la API. Se ejecuta en bucle cada 10 segundos

Aparte de la interacción principal del sistema ya presentada. En el Anexo C se presentan otras comunicaciones importantes en la aplicación. Estas son la actualización de los gráficos a partir de la actualización de datos en tiempo real y el cambio de estado del equipo actuando sobre la API.

3.2.6. Despliegue

Cuando se instala la versión de “Unity” que se quiere utilizar, se ofrecen varios complementos para añadir a la instalación que facilitan partes del desarrollo. Uno de ellos es el conjunto de herramientas (SDK) de Android, que añade automáticamente la configuración para permitir exportar aplicaciones para el sistema operativo.

A la hora de generar la aplicación, hay varias opciones que se pueden cambiar, pero basta con especificar la versión mínima de Android donde debe funcionar, el nombre y la versión de la aplicación. Con esto, Unity genera un fichero con extensión “.apk” que debe instalarse en el dispositivo móvil donde se quiera utilizar la aplicación.

Como se ve en la Figura 15, la conexión entre el dispositivo móvil que contiene la aplicación y el equipo que se monitoriza, se realiza a través de Internet utilizando una red virtual privada (VPN). Concretamente, el end-point del equipo al que se conecta la aplicación es una interfaz web (API). La conexión entre el equipo y los dispositivos que monitorizan la línea de distribución (*Analyzer* y *Grid*) se realiza mediante cables RS485 utilizando el protocolo de comunicación “Modbus”.

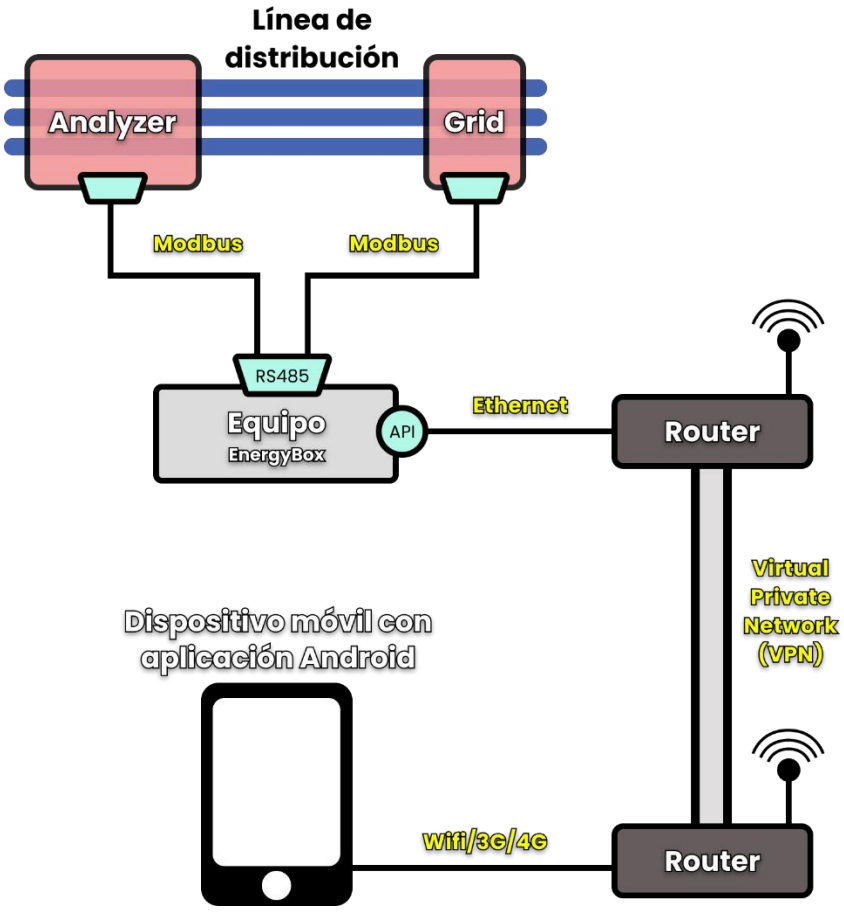


Figura 15: Diagrama de despliegue

3.3. Diseño final de la interfaz de usuario

El diseño final de la interfaz de usuario sigue el mismo patrón que en la fase de análisis. Como se muestra en la Figura 16 la aplicación divide el entorno que visualiza el usuario en tres espacios flotantes. En esta versión final se ha querido separar la funcionalidad de los espacios laterales ya que en la primera versión tanto el izquierdo como el derecho tenían como objetivo mostrar gráficos, ahora, con la inclusión de los modos de visualización del sistema, se ha reestructurado este diseño.

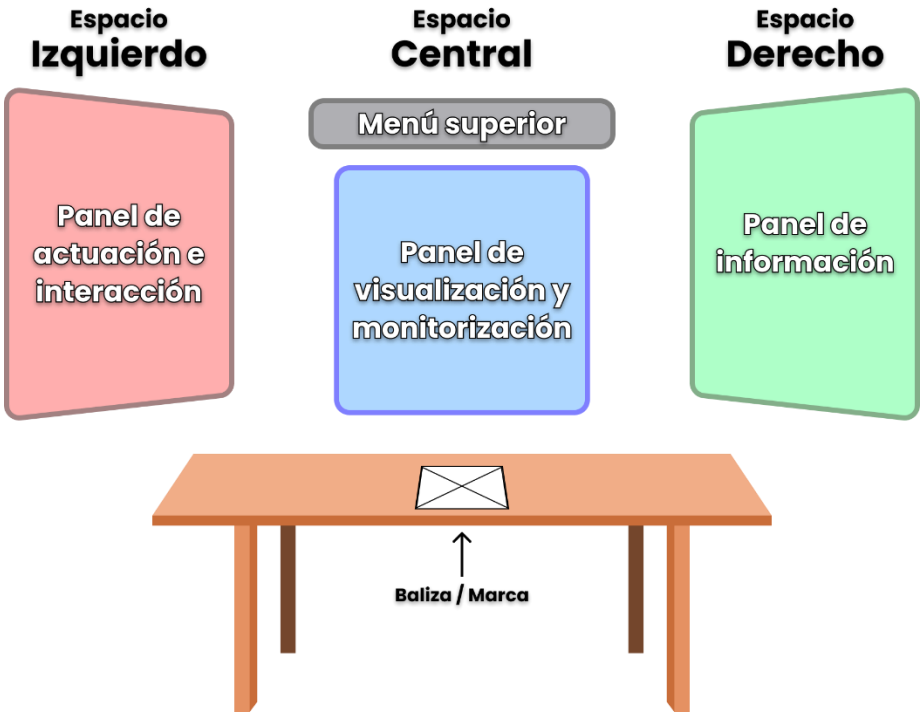


Figura 16: División por espacios de la interfaz de usuario

El espacio derecho es utilizado completamente por un panel de información. Este panel muestra los datos más relevantes del sistema y está siempre presente. El espacio central y el izquierdo están siempre relacionados y cambian dependiendo del modo de visualización que esté escogido. En el centro aparece la información importante a visualizar, como puede ser el objeto que representa el dispositivo indicando en qué estado está, o un gráfico. El izquierdo sirve para interactuar con el objeto mostrado en el panel de visualización y monitorización del centro. El espacio central cuenta también con un menú superior con el que alternar entre los distintos modos de visualización del sistema: “Monitorización del equipo” y “Gráficos”. Además, igual que se pensó en la fase de análisis, este diseño también cuenta con un menú rígido con la misma funcionalidad que el menú superior flotante, cambiar entre modos de visualización.

Espacio derecho

El espacio derecho del entorno virtual lo forma sólo un panel llamado “Panel de información”, que muestra los datos “en crudo” de las magnitudes más relevantes que mide el equipo que se está monitorizando. Estas magnitudes son: “Potencia de equipo”, “Potencia Reactiva”, “Potencia Activa”, “Voltaje” y “Corriente”. Este panel corresponde con la Vista del objeto InfoPanel, como muestra la Figura 17.

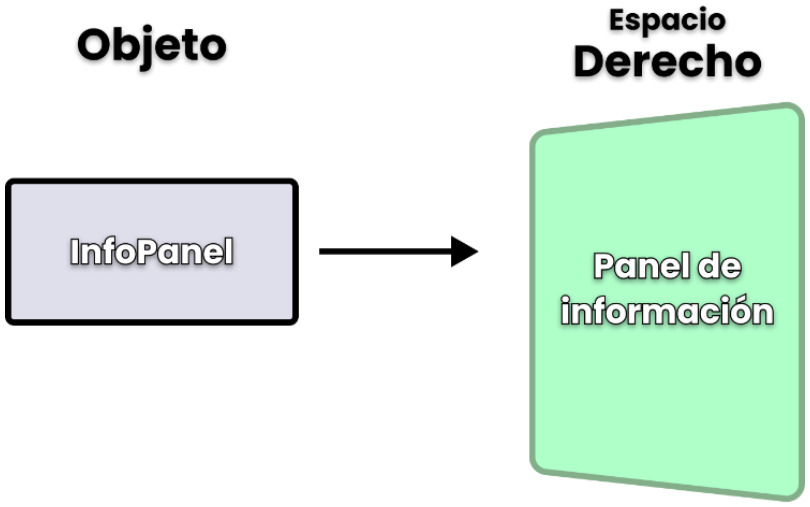


Figura 17: La vista del objeto InfoPanel corresponde con lo visualizado en el "Panel de Información" del espacio derecho

Es importante hacer énfasis en el porqué de que este panel este siempre presente y ocupe un espacio entero de visualización. A diferencia del resto, es el único que no depende del modo de visualización. El resto de los espacios muestra elementos más exhaustivos, como pueden ser el estado del equipo o un gráfico concreto, lo que dependiendo del contexto puede ser lo más importante a visualizar. Sin embargo, visualizar datos de las distintas magnitudes de forma estructurada, puede ser el complemento perfecto en cualquier otro momento en que se quiera visualizar algo más concreto. Es información importante en cualquier caso de uso de la aplicación. Por ello, se ha decidido que es una aplicación más completa para los usuarios que la van a utilizar si tienen siempre disponible la posibilidad de visualizar esta información.

Espacio central

El espacio central es el encargado de mostrar la información más valiosa al usuario. Es uno de los espacios, junto al derecho, que cambia al cambiar de modo de visualización. Está formado por el “Panel de visualización y monitorización” y por el “Menú superior”.

El “Panel de visualización y monitorización”, como su nombre indica, muestra diferentes tipos de objetos que representan el sistema y su monitorización. Depende totalmente del modo de visualización seleccionado. Igual que el espacio derecho, y como muestra la Figura 18, este espacio también se corresponde con la vista de otros objetos ya presentados, en este caso es parte de los objetos Device y Graph (la parte que solo muestra los datos).

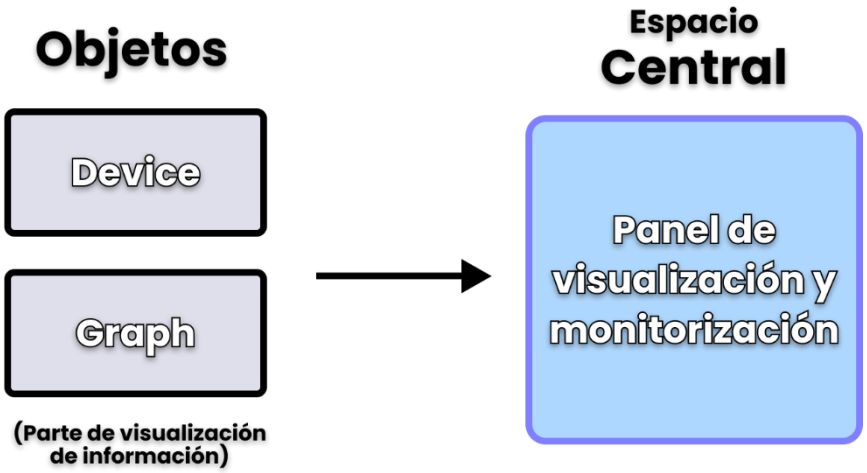


Figura 18: La parte de visualización de información de los objetos Device y Graph corresponde con lo visualizado en el "Panel de visualización y monitorización" del espacio central

Si está seleccionado el modo de “Monitorización del equipo”, este panel muestra la representación del objeto Device mediante un modelo 3D del equipo y sus líneas. Si el modo de visualización que está seleccionado es “Gráficos”, lo que se representa en este panel es objeto Graph mediante distintos gráficos.

Los objetos que se muestran en el “Panel de visualización y monitorización” son dinámicos, es decir, cada vez que la aplicación recibe una actualización de datos de parte del servidor, estos objetos deben actualizarse también sin ser necesaria una recarga de estos.

El panel “Menú superior”, que también se encuentra en este espacio, permite cambiar entre los modos de visualización disponibles en la aplicación. La Figura 19 muestra la implementación de este menú en la aplicación final.

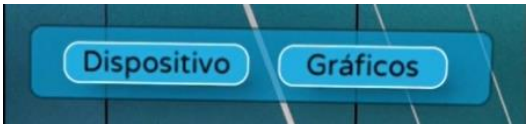


Figura 19: Menú superior

Espacio izquierdo

El espacio izquierdo de la aplicación sólo contiene el “Panel de actuación e interacción”. Al igual que el panel central, también cambia de aspecto dependiendo del modo de visualización que esté seleccionado.

Como el espacio central y como muestra la Figura 20, este espacio también muestra la Vista de los objetos Device y Graph ya presentados en el modelo de objetos, pero en este caso se muestra la parte de los objetos que realiza la actuación sobre el sistema y sobre estos propios objetos.

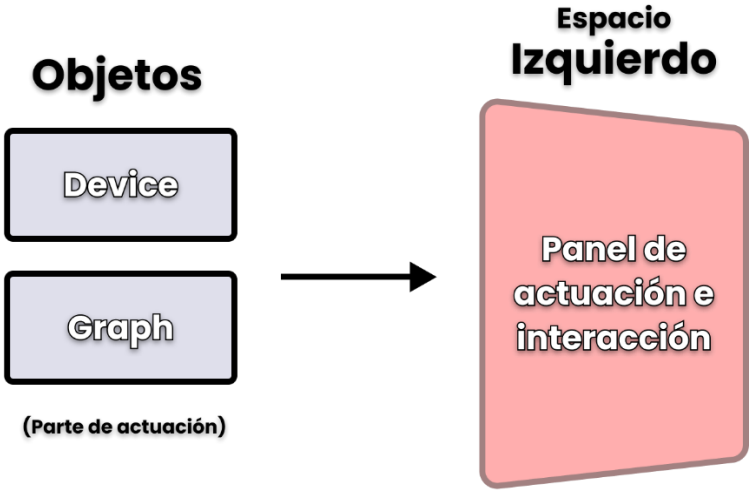


Figura 20: La parte de actuación de los objetos Device y Graph corresponde con lo visualizado en el "Panel de actuación e interacción" del espacio izquierdo

Este panel actúa sobre el estado del objeto que se está visualizando en el espacio central. Si el modo de visualización es “Monitorización del equipo”, en este espacio izquierdo aparece un panel con botones para interactuar directamente con el estado del equipo. En caso de que el modo de visualización de la aplicación sea “Gráficos”, este panel izquierdo de actuación mostrará diferentes botones para cambiar el gráfico mostrado en el panel de visualización central.

Menú rígido

El “Menú rígido” realiza la misma funcionalidad que el “Menú superior” ubicado en el espacio central. La diferencia es que para interactuar con él no hace falta enfocarlo con la cámara, ya que se encuentra siempre en la pantalla a disposición del usuario. Se puede ocultar y mostrar con los botones que se muestran en la Figura 21.

Figura 21: Menú rígido cerrado y abierto

En este apartado se han presentado los puntos más importantes de la interfaz final del usuario. Hay una explicación más extensa en el Anexo D.

4. Implementación

En este capítulo se exponen los puntos más importantes en relación a la implementación del diseño realizado en esta aplicación.

4.1. Interfaz gráfica de usuario

En el capítulo de diseño se ha comentado qué tienen que representar los objetos y cómo se tienen que comportar en la interfaz que utiliza el usuario. En este apartado se va a especificar cómo se han traslado estos objetos a modelos 3D y cómo afecta el dinamismo de la aplicación a los mismos. Todos estos modelos 3D son *GameObjects* dentro de Unity y su comportamiento se debe a la inclusión de diversos componentes.

Device

Device es la representación del equipo de control de potencia al que se conecta la aplicación. Su función principal es la de mostrar el estado en el que se encuentran el equipo y sus líneas.

Conocer el estado del equipo rápidamente es lo más importante, ya que la principal funcionalidad de la aplicación es monitorizar el propio equipo. Para ello, se ha instalado un *GameObject* con forma de cubo que representa la caja del equipo con un icono en el centro que representa el estado actual. También se han instalados otros *GameObjects,* que representan las líneas, y que indican cómo actúa el equipo sobre ellas.

BarGraph

La particularidad con la que cuenta el objeto BarGraph respecto al resto de objetos es que, en vez de haber programado un script para su control, se ha utilizado una librería descargada de Unity Asset Store, que es capaz de generar gráficos de barras dinámicos.

Hay disponible información más detallada sobre la implementación de estos objetos en el Anexo E junto con otros aspectos interesantes de la implementación de la GUI como son los botones, el texto y los iconos.

4.2. Tratamiento de datos

A partir de los datos obtenidos por el equipo, se creó el modelo de datos expuesto en el apartado 3.2.3 de este documento. Sin embargo, para mostrarlos al usuario final, se ha tenido que realizar un tratamiento de datos dependiendo del objeto y la magnitud que se tuviera que mostrar.

Panel de información y gráficos de barras

El panel de información (objeto InfoPanel) muestra los valores de las diferentes magnitudes que se monitorizan (“Potencia del equipo”, “Potencia Reactiva”, “Potencia Activa”, “Voltaje” y

“Corriente”) recogidos por los dispositivos que actúan sobre las líneas (“Analyzer” y “Grid”). La “Potencia del equipo” solo aparece en el dispositivo “Grid”, ya que es la potencia generada por la EnergyBox y el dispositivo “Analyzer” lee los valores de las antes de pasar por el equipo.

Los datos que visualiza el usuario que recibe del dispositivo “Grid” son los valores que toman las diferentes magnitudes medidas por el equipo sobre las líneas una vez se han balanceado. Estos valores son los mismos que se visualizan en los gráficos de barras de las mismas magnitudes. Por lo tanto, el tratamiento de datos es el mismo en este caso.

Aunque las medidas tomadas por el dispositivo “Grid” son realizadas una vez el equipo a equilibrado las líneas, los valores recibidos se refieren a cómo ha actuado el equipo en concreto con las cargas de las líneas. Esto quiere decir que para conocer el estado final de las líneas una vez han sido balanceadas, es necesario sumar los datos del dispositivo “Grid” con los recibidos del “Analyzer” (excepto para el voltaje).

El tratamiento de datos concreto que se realiza a partir del modelo de datos se encuentra en el Anexo F.

Gráfico de líneas - Potencia aparente

El gráfico de líneas muestra la potencia aparente [8] antes y después de que el equipo equilibre las líneas. La potencia aparente es la suma vectorial de la potencia que disipa dicho circuito y se transforma en calor o trabajo (potencia activa) y la potencia utilizada para la formación de los campos eléctrico y magnético de sus componentes (potencia reactiva). Esto significa que la potencia aparente representa la potencia total desarrollada en un circuito con impedancia Z.

El código utilizado en el script MainGraph en C# que calcula la potencia aparente es el siguiente:

```
Complex a = Mathf.Cos(2*Mathf.PI/3) + new Complex(0, Mathf.Sin(2*Mathf.PI/3));
Complex lineU = (lineU_Q + lineU_P * new Complex(0, 1));
Complex lineV = (lineV_Q + lineV_P * new Complex(0, 1)) * a * a;
Complex lineW = (lineW_Q + lineW_P * new Complex(0, 1)) * a;
Complex difference = -lineU -lineV -lineW;
```

Para calcular esta potencia en C# se hace uso de la clase Complex. Es necesaria porque se utilizan números complejos para calcular el extremo del vector que representa la potencia aparente.

La potencia aparente se calcula para las tres líneas que controla el equipo (u, v y w). También se genera otro vector con la diferencia entre estas tres líneas. En el gráfico de la aplicación esta diferencia se presenta como línea neutro y sirve para que le sea más fácil al operario analizar la diferencia de potencia entre las líneas.

4.3. Rotación de modelos 3D

Para dar al usuario algo de control sobre los objetos en 3 dimensiones, se implementó un script que permite rotar *GameObjects* sobre su eje Y, cuando el usuario arrastre el dedo por la pantalla.

Unity utiliza cuaterniones unitarios como notación matemática para representar las orientaciones y las rotaciones de objetos en tres dimensiones. Al ser una representación matemática compleja, se pueden utilizar métodos para convertir ángulos de Euler (conjunto de 3 coordenadas X, Y, Z) en cuaterniones. Esta ha sido la forma utilizada para la implementación de este script.

Utilizando la función “Update” que otorga la clase “Mono Behaviour” al script, y a partir de la distancia que recorre el dedo del usuario sobre la pantalla en cada frame, se calcula cuánto tiene que girar el *GameObject* en cada llamada a la función “Update”.

4.4. Pruebas

En general, las pruebas realizadas en la aplicación han sido manuales. Esto se debe es una aplicación centrada en la interfaz del usuario y lo que es realmente necesario es que una persona determine si lo que está observando es correcto. Aun así, también se ha implementado un test automático para verificar la conexión con la API del equipo.

4.4.1. Pruebas de usabilidad en modelos 3D virtuales

Uno de los aspectos más importantes en la usabilidad de una aplicación basada en realidad aumentada es la estabilidad y apariencia de los modelos 3D virtuales que, utilizando la cámara del dispositivo, se mezclan con el entorno real. Es importante que estos modelos estén bien situados, aparezcan con el tamaño correcto, se mantengan en su sitio ante el movimiento del usuario y que no tiemblen o parpadeen.

Vuforia provee el conjunto de herramientas que gestionan la realidad aumentada. Se han realizado varias pruebas sobre diversos puntos que pueden causar inestabilidades, tanto de reconocimiento del entorno y como de despliegue de los modelos u objetos 3D en la pantalla.

En el Anexo G se muestran las conclusiones de haber realizado dichas pruebas.

4.4.2. Test automático de conexión

Se ha implementado un test automático que realiza la conexión con la API expuesta por el equipo, recupera los datos en formato JSON y comprueba que contiene los valores necesarios para que la aplicación funcione correctamente.

Para implementar este test, se ha utilizado el lenguaje de programación C#, ya que los scripts creados para el funcionamiento de la aplicación también estaban escritos en este lenguaje, junto con el conjunto de herramientas o framework “.NET” propiedad de Microsoft. Se ha hecho uso de librerías de este framework como “Net.Http” para poder realizar una petición GET al end-point

de la API o “Diagnostics” que permite la utilización de aserciones (asserts) que realizan las comprobaciones de test requeridas.

En total, el test cuenta con tres comprobaciones o aserciones:

- Primero se comprueba que la petición GET realizada ha devuelto un JSON que se ha podido mapear al modelo de datos de la aplicación y no es nulo.
- Después, se comprueba que los valores necesarios del dispositivo Grid, que la aplicación necesita para funcionar, se encuentran en los datos recibidos previamente.
- Por último, se vuelve a realizar la misma comprobación, pero del dispositivo Analyzer con los identificadores de sus datos correspondientes.

4.4.3. Prueba manual basada en guion

Para comprobar que la aplicación funciona correctamente, la única manera es realizar una prueba manual donde se explore la aplicación al completo. Se ha creado un guion con los pasos a seguir para realizar esta prueba manual. Dicho guion se encuentra en el Anexo G.

5. Gestión del proyecto

El proyecto ha tenido una duración aproximada de 500 horas. Se inició el día 6 de octubre de 2020 y se finalizó el día 5 de marzo de 2021, contando entonces con 22 semanas de desarrollo.

Durante las dos primeras semanas del proyecto se llevó a cabo, junto al personal de las empresas involucradas, una división de las tareas por semanas. La Tabla 4 situada en la siguiente página, muestra esta planificación.

A pesar de que se planificó todo el proyecto inicialmente, se hizo de tal manera que un cambio en los requisitos no implicaba un cambio en la planificación de las tareas. Esto se debe a que desde el principio se tenía la idea clara de lo que tenía que hacer la aplicación, así como de qué tenía que incluir. Los rasgos o requisitos más generales eran la realidad aumentada, visualizar el equipo como un objeto 3D en el espacio, visualizar datos del equipo a través de paneles, crear gráficos con esos datos e interactuar con el equipo mediante consignas para cambiar su estado.

Como resumen, la planificación se puede dividir en tres partes. Cuando se empezó este proyecto no se contaba con experiencia en Unity ni en realidad aumentada, por lo que fue primordial utilizar las primeras semanas para entender su funcionamiento y empezar a desarrollar los primeros prototipos. Las primeras 7 semanas se considera que fueron la fase de análisis del proyecto (aunque ya se había empezado a idear la arquitectura de la parte de diseño), que acabaron con la presentación de la primera demo a las empresas involucradas. Durante las siguientes 11 semanas se llevaron a cabo varias demos más, ya que este tiempo se centró el desarrollo en tareas más específicas (mejora de la realidad aumentada, trabajo con los scripts, desarrollo de gráficos, interacción con el equipo, visualización de medidas en tiempo real y la implementación de una interfaz de usuario usable). Y finalmente, durante las últimas 4 semanas, se concluyó el proyecto depurando la aplicación en busca de errores, aplicando las últimas correcciones y preparando la documentación necesaria.

Se han utilizado ciertos aspectos de las metodologías ágiles en la gestión del proyecto. A través del feedback que se generaba en las reuniones periódicas de seguimiento, se avanzaba con el estudio de Unity y sus herramientas, se era capaz de observar mejor el alcance de cómo se podían implementar y especificar más los requisitos más generales comentados anteriormente.

Como conclusión, a la finalización del proyecto se considera que la gestión ha sido muy buena, ya que se han cumplido todos los requisitos previstos y el entendimiento y la sintonía entre las partes involucradas ha sido muy bueno.

Tabla 4: Planificación del proyecto

Actividades	6-10	12-10	19-10	26-10	2-11	9-11	16-11	23-11	30-11	7-12	14-12	21-12	28-12	4-1	11-1	18-1	25-1	1-2	8-2	15-2	22-2	1-3
Documentación, bibliografía: RA	X	X																				
Inicios con Unity			X																			
Inicios con Vuforia			X																			
Configuraciones y versiones				X	X	X																
Primeras pruebas RA				X	X	X																
Inicio interfaces de usuario					X																	
Inicios con C#					X																	
Inicios con gráficos						X																
Inicios con scripts para interactuar con objetos						X	X															
Ajustes y mejoras de la RA en la aplicación con Vuforia (mejorar visualización, apariencia de objetos, contacto con el entorno)								X	X													
Creación / modificación de los scripts de los distintos gráficos y objetos que se utilizan									X	X	X											
Conexión con la API (recuperación de datos)												X										
Visualización de medidas en tiempo real (crear espacios/pantallas para esas medidas)												X	X									
Desarrollo gráficos dinámicos														X	X	X						
Interacción con la API (actuación, botones para interactuar con la API)																X						
Desarrollo interfaz de usuario usable (sobre la pantalla y en realidad aumentada)																	X	X				
Modificaciones sobre la aplicación para conseguir la máxima calidad posible, solución de posibles errores existentes																			X	X	X	X
Creación de documentación sobre el trabajo desarrollado y sobre la aplicación																			X	X	X	X

6. Conclusiones y trabajo futuro

6.1. Trabajo futuro en esta aplicación

A lo largo de las reuniones de seguimiento que se han tenido, se han ideado nuevas funcionalidades que se podrían añadir a la aplicación. Estas son:

- **Implementar un nuevo modo de visualización: "Histórico de datos"**. Existe una base de datos donde se va almacenando la evolución del equipo en el tiempo. Se podría implementar este nuevo modo de visualización para mostrar por meses/semanas/días u otros métodos comparativos el cambio de las diferentes magnitudes que monitoriza el equipo, y así poder saber de forma más exhaustiva los momentos de mayor consumo en la línea de distribución.

- **Cambiar de modo automático a manual.** El equipo de electrónica de potencia puede funcionar tanto en modo automático como manual. En el modo manual el operario se encarga de mandar las consignas correspondientes al dispositivo "Grid" para balancear la línea. Por ahora la aplicación sólo conecta con el equipo en modo automático, pero sería interesante que también pudiera funcionar en modo manual y se pudieran así enviar consignas.

- **Añadir más tipos de gráficos.** En la versión actual sólo están disponibles gráficos de barras y de líneas, pero sería interesante y provechoso para los operarios ver otro tipo de gráficos.

- **Poder monitorizar varios equipos a la vez.** Sería interesante poder visualizar varios equipos a la vez y habilitar la comparación entre ellos.

- **Exportar la aplicación a otras plataformas y sistemas operativos.**

6.2. Trabajo futuro con la RA en este campo

Como se ha comentado al principio de este documento, esta aplicación es una primera aproximación a la utilización de la realidad aumentada por parte de operarios para realizar tareas de mantenimiento en equipos electrónicos.

En un futuro, la idea de las empresas involucradas es seguir avanzando con el uso de esta tecnología en este campo ya que le ven un gran potencial. Es muy interesante pensar que un operario podría arreglar o ajustar un equipo a través de las indicaciones recibidas por modelos en 3 dimensiones en realidad aumentada. Pensando también en entornos críticos donde puede haber peligros de sobretensión, aplicar realidad aumentada puede ser un factor importante y determinante para generar un entorno segur donde no haya complicaciones para los operarios.

Además, la realidad aumentada no sólo se utiliza a través de pantallas móviles o tablets: ya están apareciendo los primeros prototipos de gafas capaces de reconocer el entorno real y generar un entorno virtual en sus cristales. Esto, sin duda, es un gran avance que puede facilitar mucho el desarrollo de las actividades de gestión y mantenimiento de diferentes tipos de equipos.

6.3. Conclusiones

El resultado final de este trabajo ha sido muy satisfactorio a nivel personal. El hecho de utilizar herramientas nuevas y tecnologías en auge ha hecho que trabajar en este proyecto haya sido muy enriquecedor para mí.

Haber podido desarrollar un software destinado a solucionar un problema real ha sido un reto apasionante. Formar parte de un sistema ya implantado, que la aplicación este actualmente pasado pruebas de campo, y estar en contacto con personas de empresas involucradas en el proyecto ha hecho que experimente cómo es el mundo profesional y me ha motivado a hacer un buen trabajo.

Sin embargo, lo que me ha suscitado más interés en este trabajo ha sido la realidad aumentada y pensar en todo el potencial que tiene la fusión del entorno virtual con el mundo real. La tecnología tiene como objetivo hacer la vida más fácil a las personas y ayudarnos en nuestras tareas laborales o cotidianas. Hay muchas ideas y proyectos donde esta tecnología tiene un papel fundamental, y mezclar el entorno virtual con el real puede llevarnos a alcanzar nuevos objetivos que antes se veían muy lejanos o incluso imposibles.

La realidad aumentada ya forma parte de nuestra vida, pero en el futuro va a estar presente en todas partes.

Referencias

[1] Azuma, R. T. (1997). A survey of augmented reality. Presence: Teleoperators & Virtual Environments, 6(4), 355-385.

[2] Unity Framework. https://unity.com/es [Ultimo acceso Junio 2021]

[3] Vuforia Engine Developer Portal. https://developer.vuforia.com/ [Ultimo acceso Junio 2021]

[4] AR Foundation. https://unity.com/es/unity/features/arfoundation [Ultimo acceso Junio 2021]

[5] Visual Studio IDE. https://visualstudio.microsoft.com/es/ [Ultimo acceso Junio 2021]

[6] Documentación C#. https://docs.microsoft.com/es-es/dotnet/csharp/ [Ultimo acceso Junio 2021]

[7] Potel, M. (1996). MVP: Model-View-Presenter the Taligent programming model for C++ and Java. Taligent Inc, 20.

[8] Willems, J. L., Ghijselen, J. A., & Emanuel, A. E. (2005). The apparent power concept and the IEEE standard 1459-2000. *IEEE Transactions on Power Delivery*, *20*(2), 876-884.

Anexo A. Equipo al que se conecta la aplicación

En el apartado 2.1 se ha introducido el equipo al que se conecta la aplicación. En este anexo se añade más información al respecto.

El equipo se denomina EnergyBox y ha sido desarrollado en una Raspberry Pi v3 controlada por un sistema operativo Linux. Mediante un conector RS485 situado en la placa y utilizando el protocolo de comunicación Modbus [9] basado en Maestro/Esclavo, el equipo se comunica con dos dispositivos situados sobre la línea trifásica de distribución. El primero de estos dispositivos se denomina “Analyzer” y su objetivo es analizar la energía eléctrica que va por la línea para que, de forma automática, el equipo sepa cómo equilibrar las cargas de las fases. Los resultados de este balanceo son comunicados mediante consignas al segundo dispositivo, “Grid”, el cual aplica los cambios sobre la línea y le dice al equipo cómo han quedado finalmente las fases. En la Figura 22 se muestra el diagrama de esta conexión.

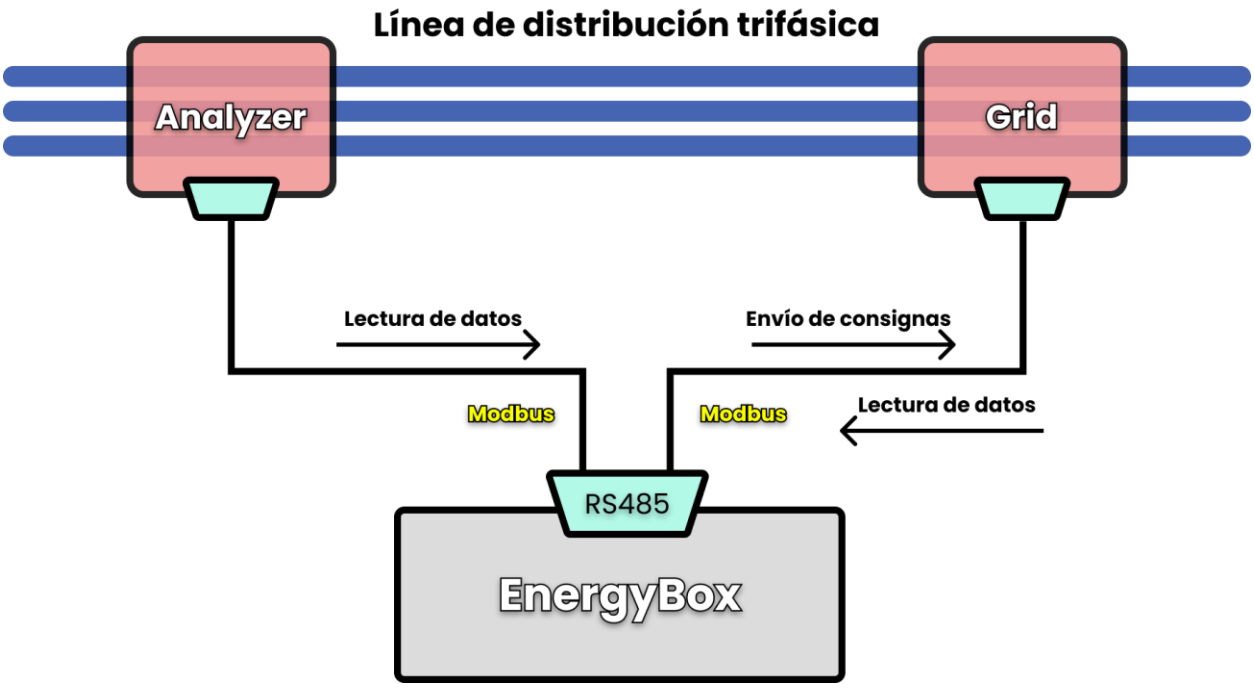


Figura 22: Conexión de la EnergyBox con los dispositivos que monitorizan y actúan sobre la red

La EnergyBox también cuenta con varios componentes que dotan al equipo con sistemas de comunicación externos adicionales. A través de un conector Ethernet, el equipo se conecta a un router que le proporciona acceso a Internet. De esta manera, el equipo expone una API web que, a través de una red privada virtual (VPN) da acceso desde el exterior al control y monitorización de la línea de distribución. La Figura 23 muestra el esquema de la EnergyBox con los componentes conectados.

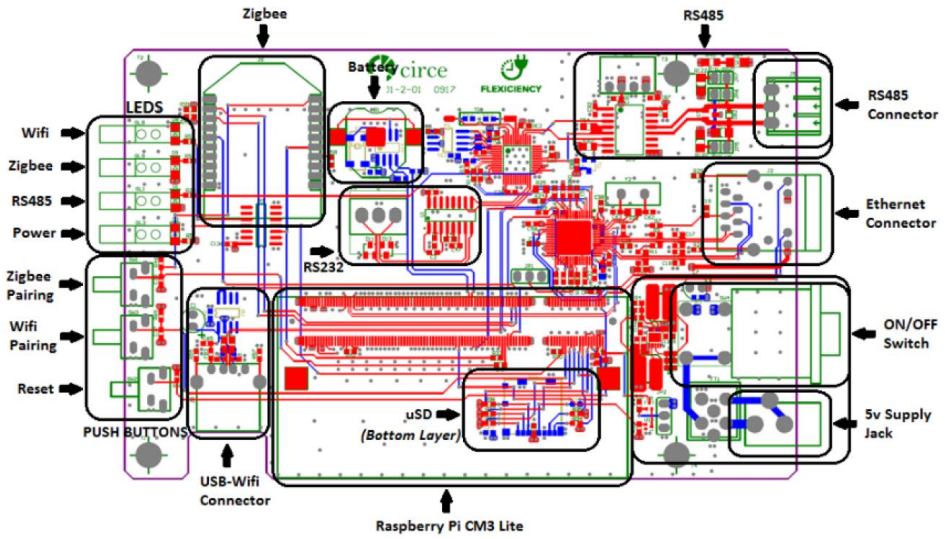


Figura 23: Diseño de la placa hardware desarrollada para el control y la monitorización de la línea

Anexo B. Arquitectura de Unity

La información de este anexo complementa lo expuesto en el apartado 3.2.1 acerca de la arquitectura de Unity.

Escenas (*Scenes*)

Las escenas son el entorno donde se sitúan los *GameObjects.* Puede haber varias escenas o, como en el caso de esta aplicación, solo una. La Figura 24 muestra cómo se ve una escena vacía en Unity.

A la hora de crear una escena, esta viene con dos *GameObjects* ya incluidos, una cámara y una luz. A partir de aquí, estos se pueden configurar como se requiera en cada momento y añadir más *GameObjects*.

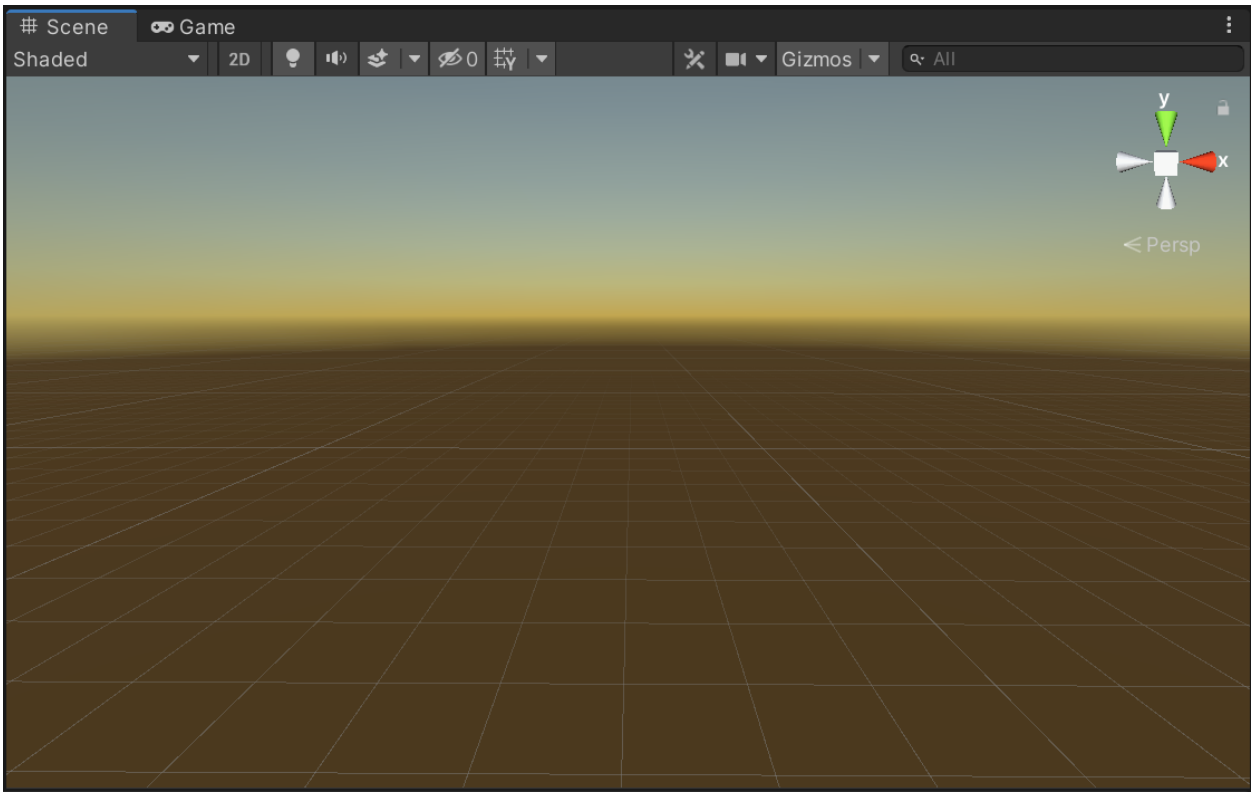


Figura 24: Vista de una escena en Unity

GameObjects

Los *GameObjects* son el concepto más importante dentro de la edición de una aplicación en Unity, son los elementos que representan todo lo que hay dentro de una escena. La Figura 25 muestra un ejemplo de escena con varios *GameObjects* en ella. Por ejemplo, la cámara y la luz que se crean junto a cada nueva escena son *GameObjects,* pero también lo son un cubo, una esfera o un árbol que se utilizan para adornar un entorno gráfico.

Un *GameObject* no hace nada por sí solo, necesita de propiedades para poder convertirse en algo en específico. Estas propiedades se otorgan agregando components. Según el tipo de objeto que se desee, habrá que agregar diferentes tipos de combinaciones de components.

Los *GameObjects* pueden contener otros *GameObjects* y así crear objetos más complejos. Puede darse el caso de que haya *GameObjects* vacíos (sin componentes, es decir, sin propiedades y sin funcionalidad) que simplemente sean utilizados como contenedores para agrupar otros *GameObjects*.

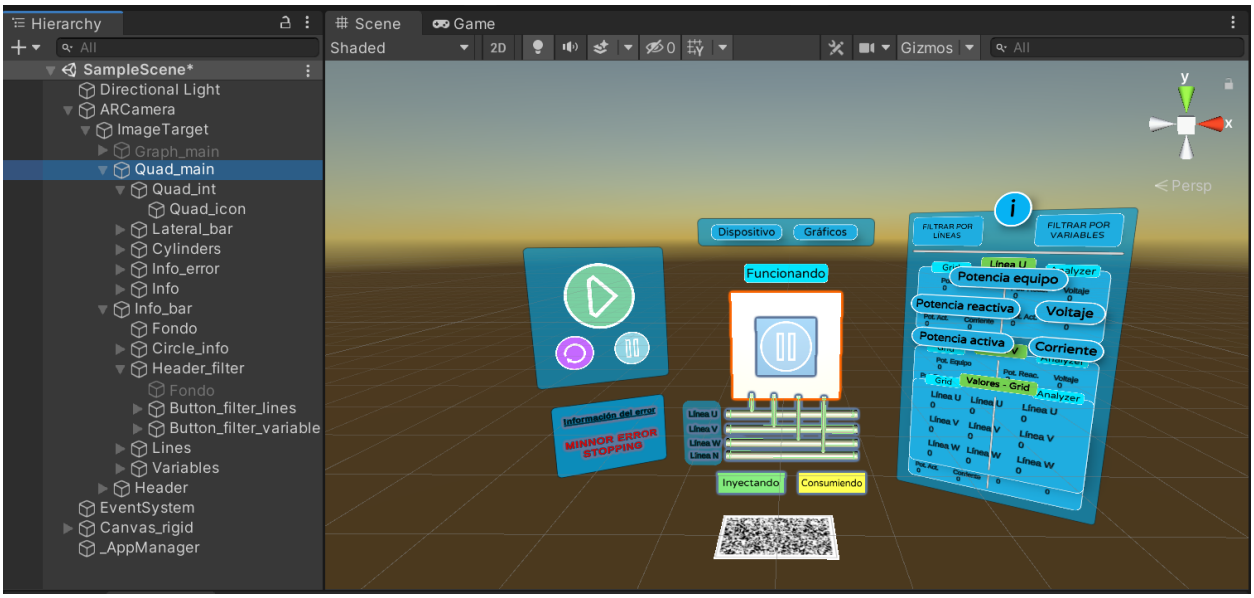


Figura 25: Vista de una escena con GameObjects y jerarquía de los GameObjects

Componentes (*Components*)

Los componentes son los elementos del sistema Unity que otorgan propiedades y funcionalidad a un *GameObject*, es decir, definen su comportamiento. La Figura 26 muestra un *GameObject* con varios componentes. Podemos decir que los componentes son funcionalidad aislada que se va agregando a los *GameObjects* de forma que se van creando diferentes objetos dependiendo de las combinaciones de components que hemos agregado.

Unity cuenta con varios componentes diferentes ya creados. El más común es transform, con él se indica la posición del *GameObject* en el espacio de la escena. Otro ejemplo sería que dependiendo de cómo se quiere mostrar el objeto, se agregará un componente u otro de renderizado (*Mesh-Renderer*, *Particle-Renderer*, *Line-Renderer*…).

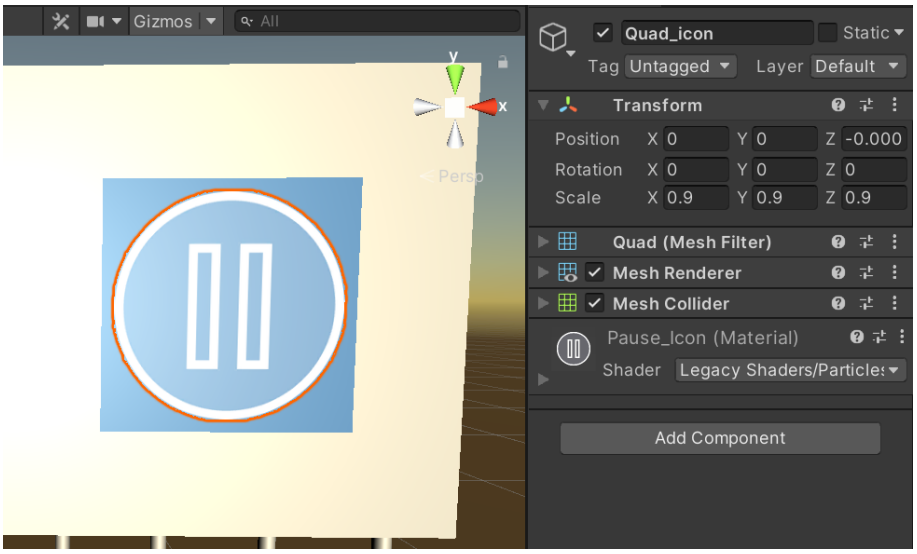


Figura 26: GameObject con Componentes asociados

Si se quiere añadir una propiedad o funcionalidad a un *GameObject* que no está incluida en Unity, siempre se puede crear un componente personalizado a través de un script.

Un script es un programa que implementa una nueva propiedad personalizada por el propio desarrollador para ampliar las funcionalidades de los objetos ya que Unity interpreta los scripts como componentes (como muestra la Figura 27). De esta manera, el programador puede desarrollar sus propios comportamientos de los *GameObjects*.

Unity soporta como lenguaje nativo C# aunque también permite la utilización de otros lenguajes del framework .NET. En el caso de esta aplicación, sólo se ha utilizado C#.

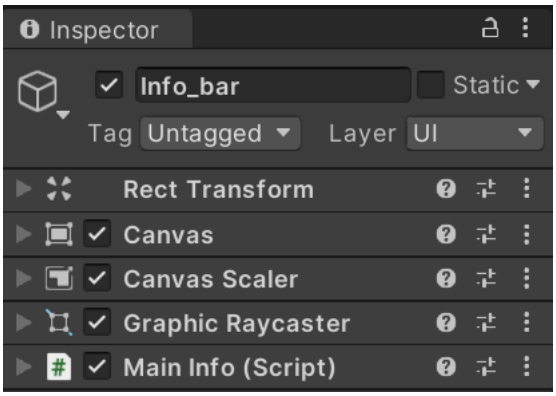


Figura 27: Componentes de un GameObject con un Script

MonoBehaviour

MonoBehaviour es la clase base de la cual, por defecto, derivan todos los scripts y componentes de Unity. Cuando se crea un script desde Unity Editor, este automáticamente hereda de MonoBehaviour y añade unos métodos por defecto. La Figura 28 muestra un script generado desde Unity con la clase Mono Behaviour.

Esta clase proporciona al script acceso a una colección de eventos los cuales permiten ejecutar el código en base a lo que está ocurriendo actualmente en el proyecto. Los eventos más comunes son “start”, el cual se lanza solo una vez al instanciar el objeto al que está unido el script como componente, y “update”, el cual se lanza con cada actualización de los frames que se producen en la aplicación.

Utilizando estos dos eventos es posible crear videojuegos o aplicaciones con Unity. Sin embargo, otra función interesante que integra la clase MonoBehaviour son las corrutinas (*coroutines* [10]). Cuando se lanza cualquier función en un script de Unity, esta se ejecuta completamente antes de actualizar el siguiente cambio de frame, mientras que una corrutina es una función que tiene la habilidad de pausar su ejecución y devolver el control a Unity para luego continuar donde lo dejó en el anterior frame. Muchas tareas en un juego o una aplicación deben llevarse a cabo periódicamente y la forma más obvia de hacerlo es incluirlas en la función Update. Sin embargo, esta función generalmente se llamará muchas veces por segundo. Cuando una tarea no necesita repetirse con tanta frecuencia, se puede incluir en una corrutina para obtener una actualización automática y regular, pero no en todos los frames. Pueden ser utilizadas como una forma de difundir un efecto durante un período de tiempo, pero también es una optimización útil. El ejemplo donde se ha aplicado este tipo de funciones en esta aplicación es en la lectura de los datos en la API donde interactúan dos corrutinas entre sí, la primera espera un número de segundos establecido entre petición y petición sin que le afecte el número de frames y la segunda realiza la llamada a la API parando la ejecución de ese hilo mientras espera la respuesta del servidor.

```
public class script : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

Figura 28: Script inicializado con MonoBehaviour

Vuforia

Vuforia es la biblioteca que se ha utilizado para la integración de la realidad aumentada en Unity.

Vuforia abstrae al desarrollador de la mayor parte de la complejidad que requiere el uso de la realidad aumentada, sin embargo, sí que es posible configurar varios aspectos para adaptar la utilización de esta tecnología a lo que la aplicación en concreto necesita. Por ejemplo, el reconocimiento de la baliza y del espacio es un punto importante para que los objetos virtuales, que se muestran a través de la pantalla del dispositivo fusionados con el entorno real, sean lo más estables posible y en caso de que la cámara pierda la marca de su foco de visión, la aplicación siga funcionando sin problema al haber realizado previamente con éxito el reconocimiento del entorno.

La integración de Vuforia en Unity se realiza de la siguiente manera. Tras instalar la biblioteca en el proyecto con su conjunto de herramientas e introducir la clave que generan exclusiva para el desarrollador, en el editor de Unity aparece la posibilidad de introducir GameObjects de Vuforia. En este caso hacen falta solo dos: un objeto cámara que lleva ya introducidos los components necesarios para hacer funcionar la realidad aumentada; y un Image Target, el cual representa la baliza que la aplicación tiene que reconocer para mostrar la aplicación a partir de la misma. Todos los objetos que se quieren mostrar tienen que derivar de este Image Target.

Plugins

Conjunto de scripts creados por otros usuarios que forman una librería o biblioteca para añadir más funcionalidad. Unity cuenta con una tienda llamada “Unity Asset Store” donde se encuentran una gran cantidad de plugins, algunos de ellos gratuitos. Es muy sencillo integrarlos en Unity al comprarlos en su tienda.

Anexo C. Comunicación y dinámica del sistema

En el apartado 3.2.5 se presenta la interacción principal del sistema. Esta consiste en la actualización de datos en tiempo real. En este anexo se muestran otras comunicaciones importantes en la aplicación.

Actualización de gráficos

El objeto Graph es quien controla todo lo relativo a los gráficos. Se comunica con AppManager para pedir una nueva actualización de los datos, y cuando los recibe los trata y se los envía a BarGraph o a LineGraph según el gráfico que se esté representando.

Es importante destacar que solo se muestra un gráfico a la vez y dependiendo de qué tipo de gráfico se esté visualizando (barras o líneas) o qué magnitud esté elegida, el tratamiento de datos que hace el objeto Graph es diferente. Cada vez que hay una actualización de los datos se comprueba que gráfico en concreto se quiere visualizar y se realiza el tratamiento correspondiente. Esto implica que cada vez que se produzca un evento en la aplicación que derive en cambiar la magnitud del gráfico que se está visualizando o el propio tipo de gráfico, se tengan que tratar de nuevo los datos. Ya que el objeto AppManager es el que contiene la variable con estos, cada vez que se produce este evento hay que realizar la petición de la variable.

Algunos *GameObjects* que contiene el objeto Graph pueden provocar este evento y comunicar a Graph que es necesario actualizar, como sucede en el caso de los botones.

El primero de los diagramas de secuencia, disponible en la Figura 29, muestra algunas de las interacciones que provocan un evento en el objeto Graph para que se actualicen los gráficos. Estas interacciones pueden ocurrir:

- Al llegar una nueva respuesta desde el servidor con una actualización de los datos.
- Al pulsar un botón para cambiar el tipo de gráfico.
- Al pulsar un botón para cambiar el tipo de magnitud a observar en el gráfico.

Los botones aparecen dibujados en el diagrama como objetos, ya que son *GameObjects* contenidos dentro del objeto Graph.

Como muestra el segundo diagrama de secuencia (disponible en la Figura 30), una vez recibido el evento, el objeto Graph pide a AppManager la última versión de los datos, realiza su tratamiento según qué gráfico en concreto se quiera mostrar y manda los datos necesarios a BarGraph o a LineGraph (según el que esté activo en ese momento).

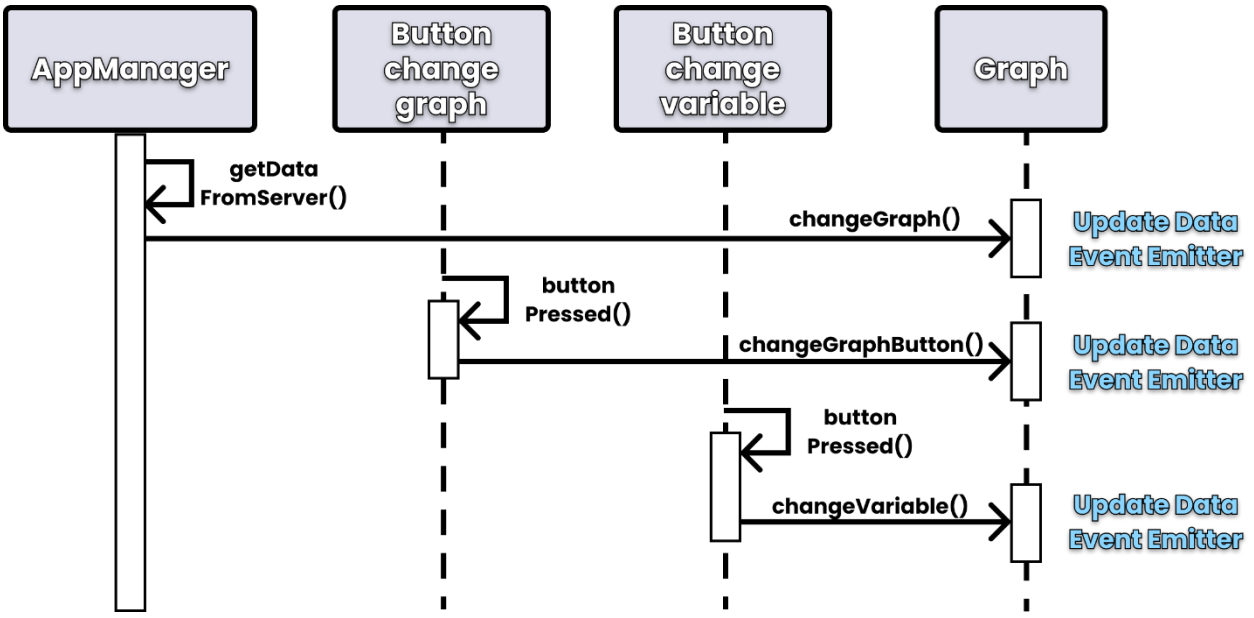


Figura 29: Diagrama de secuencia que muestra las diferentes interacciones en la aplicación que generan un evento de actualización de datos sobre el objeto Graph

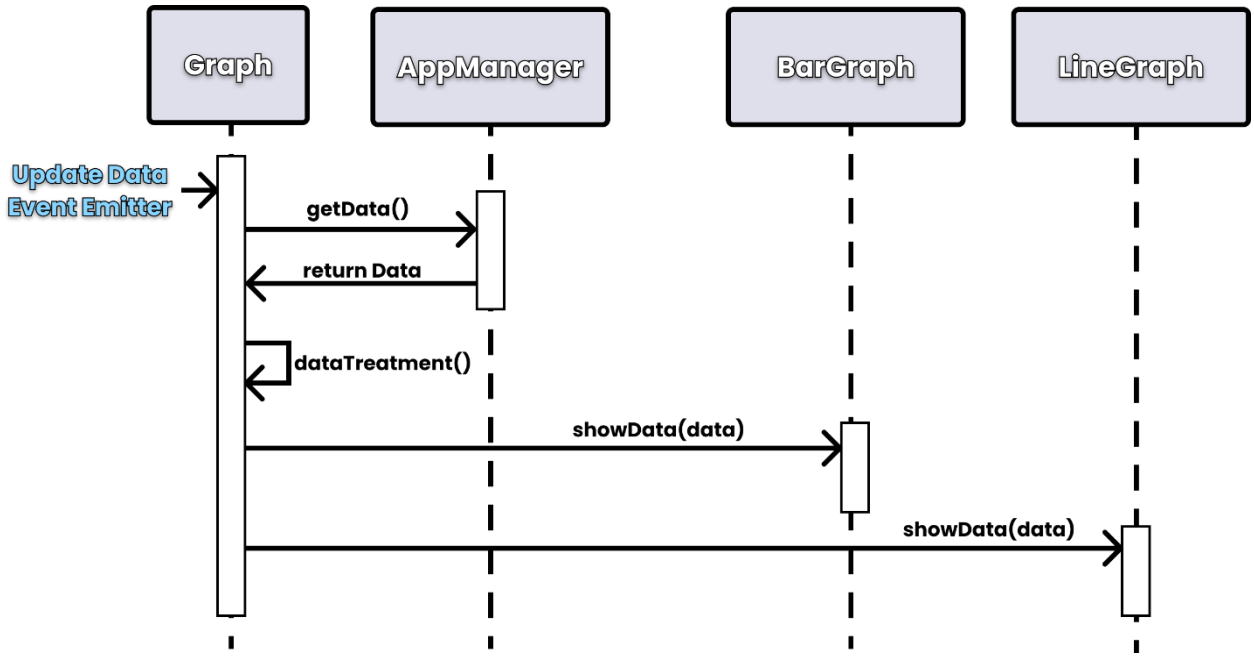


Figura 30: Diagrama de secuencia que muestra cómo se realiza la actualización de los datos a través del objeto Graph cuando este recibe un evento

Cambiar el estado del equipo actuando sobre la API

En este apartado se va a explicar cómo actúa la aplicación sobre el equipo, pudiendo cambiar el estado en el que se encuentra. Como se ha explicado anteriormente, el objeto AppManager es el encargado de realizar la comunicación con el equipo.

El objeto encargado de monitorizar los datos y exponer al usuario la posibilidad de cambiar el estado del equipo es Device. Este objeto muestra un panel de actuación con tres botones que, mediante iconos y diferentes colores, indican al usuario que se puede poner el equipo en los estados “Funcionando”, “Pausa” y “Reinicio”. Estos botones son *GameObjects* que pertenecen al objeto Device. Cuando uno de estos botones es pulsado, se lo comunican a MainDevice (el script de Device) y éste a AppManager para que envíe la petición correspondiente a la API del dispositivo mediante la creación de una corrutina. Si la API responde con un “OK”, AppManager lo comunicará al objeto Device y el modelo 3D del objeto cambiará adaptándose al estado escogido; si ocurre algún error, el equipo notificará de qué error se trata y la aplicación lo reflejará en el modelo 3D.

El diagrama de secuencia de la Figura 31 a) muestra esta interacción entre los objetos. En el diagrama se incluyen los tres botones, que son *GameObjects*, y que permiten cambiar el estado como un solo componente, ya que con los tres se realiza la misma comunicación (solo cambia el identificador pasado como parámetro). Estos botones se corresponden con los que aparecen en la Figura 31 b) respecto a la interfaz de usuario.

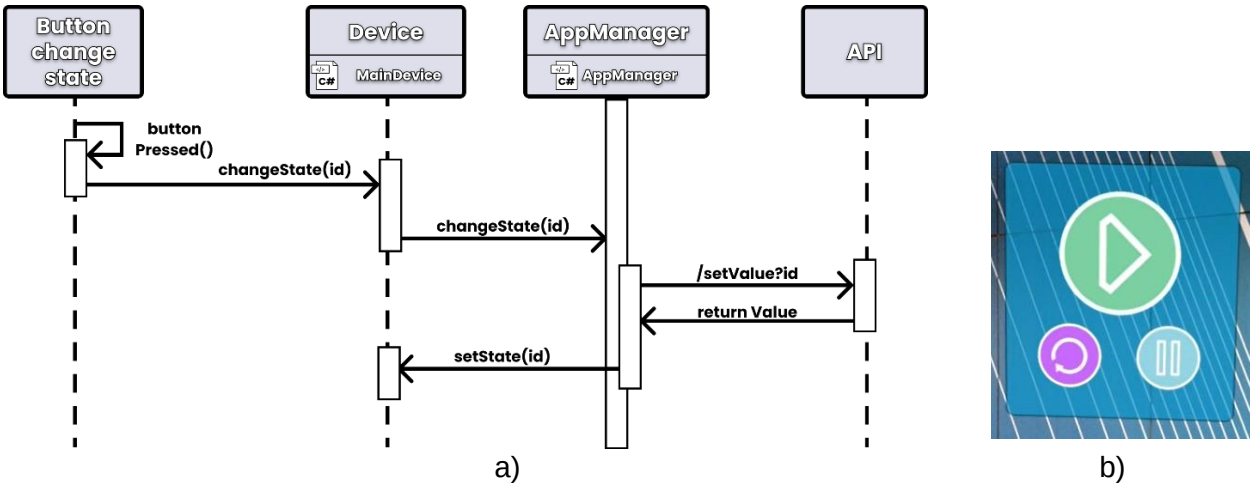


Figura 31: a) Diagrama de secuencia que comunica un cambio de estado al equipo y muestra el resultado
b) Botones que provocan un cambio de estado del equipo

Anexo D. Diseño final de la interfaz de usuario

En el apartado 3.3 se ha introducido el diseño final de la interfaz de usuario. En este anexo, se complementa la información del apartado mencionado dividido por espacios (derecho, central e izquierdo).

Espacio derecho

Los datos que se muestran en este panel están divididos en dos apartados, “Grid” y “Analyzer”. “Analyzer” hace referencia a las medidas de estas magnitudes antes de pasar por el equipo, es decir, antes de que el equipo balancee las cargas. El apartado de “Grid” es la suma entre los datos medidos en el “Analyzer” más la suma de los que suministra o consume el equipo de cada línea, es decir, son los datos finales una vez el equipo ha estabilizado las líneas. Gracias a esto, los operarios pueden rápidamente ver el resultado de cómo el equipo actúa sobre las líneas. La “Potencia de equipo” es la única que no se muestra en la sección “Analyzer” ya que es la utilizada por el equipo para funcionar y actuar sobre las líneas.

Este panel permite el filtrado de información por líneas o por variables. Al seleccionar la primera opción, el panel muestra la información de cada una de las magnitudes por cada una de las líneas que se monitorizan, es decir, muestra todos los datos a la vez; además, mediante el uso de colores, se representa sobre qué línea el equipo está consumiendo energía y sobre cual está inyectando. La segunda opción de filtrado, filtro por variable, es una forma más compacta de visualización de información, ya que sólo muestra los resultados de una variable en concreto, de esta forma se otorga la posibilidad al usuario de visualizar solo la magnitud que requiera en ese momento. La Figura 32 muestra el resultado final de este panel de información.

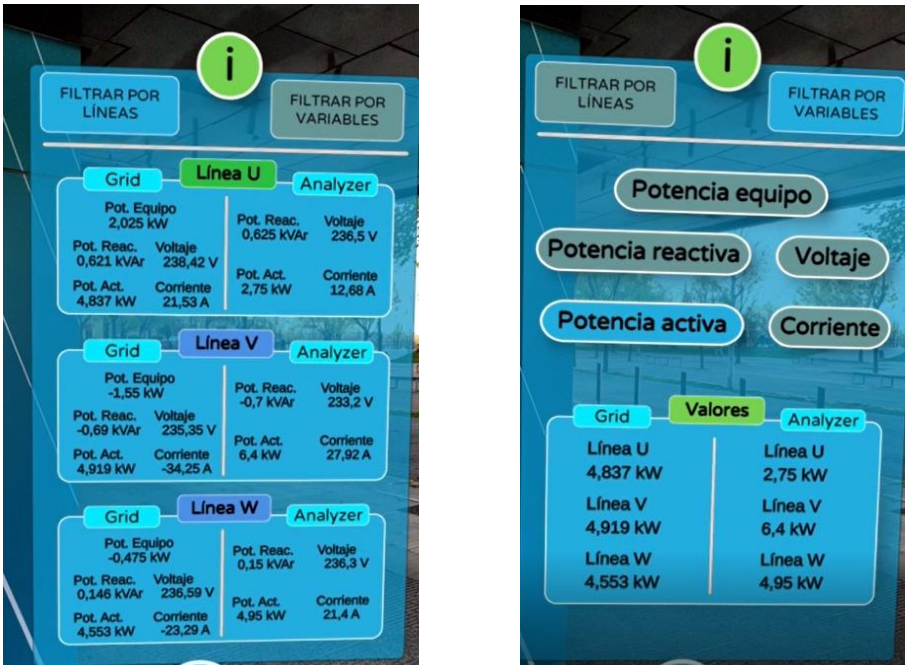


Figura 32: Panel de información filtrado por líneas y por variables

Espacio central

El espacio central es el encargado de mostrar la información más valiosa al usuario. Es uno de los espacios, junto al derecho, que cambia al cambiar de modo de visualización. Está formado por el “Panel de visualización y monitorización” y por el “Menú superior”.

Si está seleccionado el modo de “Monitorización del equipo”, este panel muestra la representación del objeto Device mediante un modelo 3D del equipo y sus líneas. En el centro del equipo, mediante un icono, se informa del estado en el que está en el dispositivo (aparte de aparecer encima en formato de texto); en las líneas, mediante un código de colores que se representa en una leyenda, se representa sobre qué líneas el equipo está consumiendo o inyectando energía (siempre que el equipo este en estado “Funcionando”). En la Figura 33 se pueden observar varias imágenes de este panel.

Figura 33: Panel de visualización y monitorización con diferentes estados del equipo (modo de visualización “Monitorización del equipo”)

Si el modo de visualización que está seleccionado es “Gráficos”, lo que se representa en este panel es objeto Graph mediante distintos gráficos. Hay dos tipos de gráficos que la aplicación puede mostrar, de barras o de líneas. El gráfico de barras compara en una dimensión los valores por línea de una magnitud en concreto. El gráfico de líneas muestra en dos dimensiones la “Potencia aparente” del equipo. En la Figura 34 se muestran imágenes de estos gráficos.

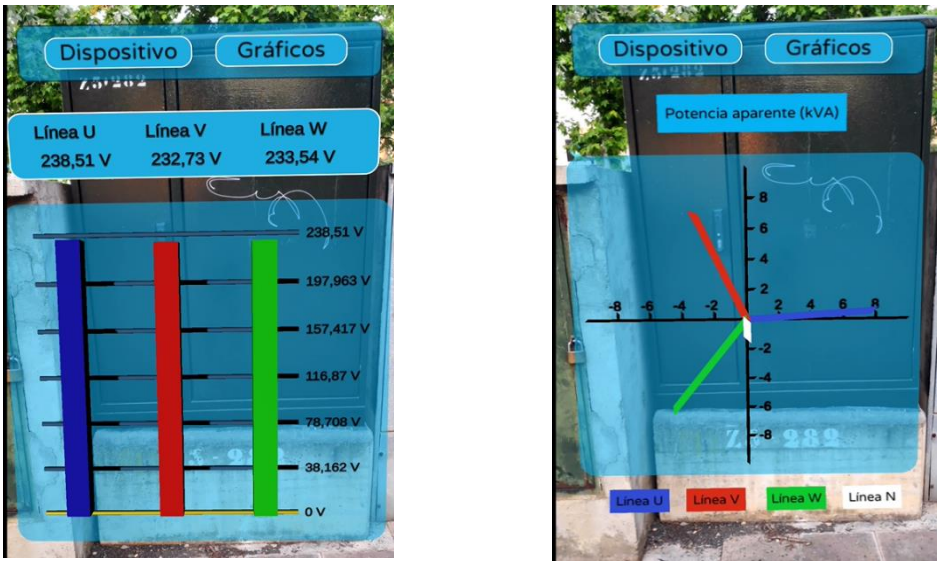


Figura 34: Panel de visualización y monitorización mostrando gráficos de líneas y de barras (modo de visualización "Gráficos")

Los objetos que se muestran en el "Panel de visualización y monitorización" son dinámicos, es decir, cada vez que la aplicación recibe una actualización de datos de parte del servidor estos objetos deben actualizarse también sin ser necesario una recarga.

Espacio izquierdo

El espacio izquierdo de la aplicación sólo contiene el "Panel de actuación e interacción". Al igual que el panel central, también cambia de aspecto dependiendo del modo de visualización que esté seleccionado.

Este panel actúa sobre el estado del objeto que se está visualizando en el espacio central. Si el modo de visualización es "Monitorización del equipo", en este espacio izquierdo aparece un panel con botones para interactuar directamente con el estado del equipo. Se puede alterar el estado a "Funcionando", "Pausa" y "Reinicio" utilizando los botones que aparecen en la Figura 35, en caso de que el equipo esté en error, aparecerá un pequeño panel debajo de este indicando que error en concreto está sufriendo el equipo mediante el código del error y un texto explicativo.

Figura 35: Panel de actuación para cambiar el estado del equipo

En caso de que el modo de visualización de la aplicación sea “Gráficos”, este panel izquierdo de actuación mostrará diferentes botones para cambiar el gráfico mostrado en el panel de visualización central. Consta de dos botones para cambiar entre gráficos de barras y gráficos de líneas, si está seleccionado el gráfico de barras se podrá seleccionar una magnitud para mostrar en el gráfico entre estas: “Potencia de equipo”, “Potencia Reactiva”, “Potencia Activa”, “Voltaje” y “Corriente” (como se puede observar en la Figura 36). Si está seleccionado el gráfico de líneas, desde este panel izquierdo se podrá elegir la representación de la “Potencia aparente” en el gráfico mediante los datos medidos antes de que el equipo balancee las cargas (medidos por el “Analyzer”) y los datos medidos después de que se produzca este balanceo (sumados los del “Analyzer” con el “Grid”), de esta forma se puede visualizar claramente el impacto del equipo en las líneas.

Figura 36: Panel para cambiar la magnitud y el tipo de gráfico a visualizar

Anexo E. Detalles de la implementación de la GUI

En el apartado 4.1 se han comentado detalles de la implementación de la interfaz gráfica de usuario para el objeto Device y BarGraph. En este anexo se completa esa información y se comentan puntos importantes en la implementación final sobre los botones, texto e iconos.

Device

Device es la representación del equipo de control de potencia al que se conecta la aplicación. Su función principal es la de mostrar el estado en el que se encuentran el equipo y sus líneas.

Conocer el estado del equipo rápidamente es lo más importante ya que la principal funcionalidad de la aplicación es monitorizar el propio equipo. Como se observa en la Figura 37, se ha instalado un *GameObject* con forma de cubo que representa la caja del equipo con un icono en el centro que representa el estado actual. Para añadir información textual al icono hay otro *GameObject* encima en forma de texto que también indica el estado.

Figura 37: Modelo 3D del equipo cuando se encuentra en el estado "Funcionando"

Cuando el equipo está en el estado “Funcionando”, consume energía de una o varias líneas y la inyecta en el resto para equilibrar las cargas. Estas líneas están representadas mediante *GameObjects* en forma de tubos que cambian de color según si el equipo está consumiendo o inyectando energía en ellas como se observa en la Figura 38.Hay una leyenda de colores para indicarle esto al usuario. La línea neutro se representa de color blanco. Cuando el estado del equipo es “Pausa, “Reiniciando” o “Error”, las líneas siempre tienen el mismo color.

Figura 38: Representación de las líneas

BarGraph

La particularidad con la que cuenta el objeto BarGraph respecto al resto de objetos es que, en vez de haber programado un script para su control, se ha utilizado una librería descargada de Unity Asset Store que es capaz de generar gráficos de barras dinámicos.

Al inicio del proyecto se planteó la posibilidad de utilizar plugins que facilitaran el desarrollo y la idea de utilizarlos para los gráficos fue la más comentada. Se buscó en la tienda de Unity algún plugin que creara gráficos de barras en 3D y se acabó eligiendo la opción “3D Interactive Barchart” [11] de la empresa “Viitorcloud Technologies” debido a que es una librería gratuita y parecía un software sencillo de utilizar y de integrar en la aplicación.

La licencia que incluye ViitorCloud en este plugin permite su modificación y su integración en cualquier proyecto siempre que se les referencie.

Una vez se obtuvo la librería, se estudió todo el material que contenía, los scripts que permiten la creación y el control de los gráficos, los materiales que se utilizan para dar estilos diferentes, imágenes y escenas de ejemplo que se incluyen junto al manual de usuario para aprender más rápido a utilizarla. Después, se integró en la escena de esta aplicación y se empezaron a realizar modificaciones en los scripts y demás material incluido para adaptar el plugin a las necesidades del proyecto. Este conjunto de scripts y sus modificaciones forman el objeto BarGraph del sistema.

Se realizaron varias modificaciones:

- En el script donde se genera la instancia del gráfico, vienen incluidos algunos métodos que generan conjuntos de datos aleatorios que sirven de ejemplo para visualizar como cambian las barras del gráfico dinámicamente. Sin embargo, no había ningún método ya escrito para actualizar el gráfico dándole un conjunto de datos definido en otra parte por lo que se creó. De esta forma el objeto Graph, cuando realiza el tratamiento de datos adecuado que se requiere visualizar en el gráfico, invoca el nuevo método creado en BarGraph pasándole como argumento el conjunto de datos que se ha calculado.

- El modelo 3D del gráfico incluye unos planos en los 3 ejes X, Y y Z que sitúan las barras del gráfico y ayudan a dar mayor sensación de profundidad. Estos planos no funcionan muy bien a la hora de cambiar el tamaño y la separación entre las barras del gráfico ya que cuentan con varios problemas de escalabilidad. Se decidió eliminarlos y crear un panel liso que se sitúa detrás de las barras del gráfico para mejorar la visualización de este.

- Para indicar mejor los valores de las barras, a lo largo de los ejes del gráfico se incluyen esferas con textos que sirven de puntos de referencia, pero tienen el mismo problema que los planos de los ejes. Cuando se cambia el tamaño del gráfico, estos puntos no escalan bien y algunas de las esferas desaparecían y otras se hacían mucho más grandes. Se decidió quitar estos puntos de referencia y crear unas líneas de referencia sobre el plano liso también creado y así tener más control sobre ellos.

Cómo este gráfico tiene que mostrar valores de cinco magnitudes diferentes, se valoró crear cinco instancias de este gráfico e ir ocultándolas y mostrándolas según el que fuera necesario visualizar. Al realizar alguna prueba ocurrieron bastantes problemas al intentar actualizar gráficos que estaban ocultos ya que Unity pierde la referencia de los *GameObjects* que no se están mostrando, por lo que se decidió que lo mejor sería tener un solo gráfico e ir actualizando sus valores.

La mayoría de las magnitudes que se representan en este gráfico pueden tener valores positivos o negativos en las diferentes líneas. Esto es un problema ya que la librería solo permite construir la barra del gráfico empezando desde el origen del eje X, no desde la mitad. Después de intentar varias soluciones, se decidió implantar dos instancias del gráfico, uno para la mitad superior y otro, girado 180º en el eje Y, para la mitad superior. De esta manera se consiguió el efecto esperado, parece un solo gráfico donde el valor 0 está situado en la mitad del eje X pero en realidad son dos gráficos girados. Esta solución requiere enviar un conjunto de valores diferente a cada una de las instancias de los gráficos, cuando el objeto Graph realiza el tratamiento de los datos, al gráfico superior le envía los datos positivos y 0 en los datos negativos y, al contrario, para el gráfico inferior.

Durante el desarrollo de los gráficos, se tuvieron en cuenta varias funcionalidades que finalmente no se llegaron a implementar. Una de ellas fue que las barras del gráfico cambiaran de color según si los datos alcanzaban algún límite peligroso ya fuera por valores muy altos o bajos. Se descartó ya que dependiendo del contexto las magnitudes pueden variar bastante y se decidió que simplemente se mostrara un color por línea y fuera el propio operario el que determinara si el nivel de peligrosidad de cada valor. Además, estos gráficos también ofrecen diversos eventos al pulsar sobre las barras. No han sido implementados en la aplicación ya que no se encontró una gran utilidad para ello.

Al principio también se planteó la utilización de gráficos circulares o también llamados gráficos de tarta (directamente traducido de la palabra inglesa *Pie Charts*) ya que “ViitorCloud Technologies”, la misma compañía del plugin utilizado para los gráficos de barras, también tiene una librería para este gráfico. Se llegó a probar en alguna demo, pero al final se descartó debido a que no se le encontró un gran beneficio a su uso con los datos que se querían mostrar.

Botones

Para darle más dinamismo a la aplicación, se valoraron diversas librerías que incluían animaciones en algunos aspectos como al esconder o hacer aparecer modelos 3D o efectos al interactuar con la aplicación a través de la pantalla.

Se decidió utilizar el plugin “Lean GUI” [12] ya que incluye varias animaciones que pueden hacer la aplicación más atractiva al usuario. Esta librería ha sido utilizada para generar las animaciones de los botones al ser pulsados. Estos botones incluyen el componente llamado “Lean Button” que les da la posibilidad de crear transiciones como cambiar de un color a otro al pasar de activos a inactivos (y viceversa) o sensación de profundidad al ser pulsados y mantenidos. Este plugin también incluye otras funcionalidades que finalmente no fueron implementadas como *radio buttons*, *toggle buttons*, mensajes modales, elementos que se pueden arrastrar por la pantalla o incluso *joysticks*.

Texto

Para la mayoría de los textos que aparecen en la aplicación se ha utilizado la librería “TextMeshPro”. Esta librería viene incluida en muchos de los plugins que Unity pone a disposición de los usuarios en su tienda “Unity Asset Store”.

Esta librería está compuesta por un componente llamado “TextMeshPro - Text” que se acopla a un *GameObject* otorgándole más estilos que un componente “Text” básico de Unity. Este componente añade funcionalidades extra al texto como:

- Espaciado de carácter, palabra, línea o párrafo.
- Texto justificado.
- Enlaces.
- Uso de diversas fuentes y materiales en el mismo campo de texto.
- Estilos personalizados.
- Sombras.
- No se necesita estar sobre un *canvas* para mostrar el texto.
- Aunque añade muchas más funcionalidades, el texto es igual de pesado que utilizando el componente “Text” normal.

Iconos

Para los iconos que representan el estado del equipo y los símbolos sobre los botones que actúan sobre este, se buscó una librería de imágenes (*sprites*) que concordara con la aplicación que se estaba desarrollando. Tras realizar esta búsqueda se eligió el plugin “Clean Vector Icons” [13] disponible en “Unity Asset Store” ya que cuenta con una gran variedad de iconos y contiene todos los necesarios para esta aplicación.

Anexo F. Fórmulas utilizadas para el tratamiento de datos

En el apartado 4.2 se ha expuesto el tratamiento de datos que se realiza para mostrar los datos finales a los usuarios que utilizan la aplicación. En este anexo se muestran las fórmulas utilizadas para el tratamiento concreto de los datos.

Los datos que el usuario debe observar en el panel de información (objeto InfoPanel) sobre el dispositivo “Analyzer” no sufren demasiadas modificaciones. El tratamiento concreto se muestra en la Tabla 5.

Tabla 5: Tratamiento de datos realizado para el objeto InfoPanel a través de los datos recibidos por el dispositivo Analyzer

Magnitud	Conversión realizada desde el modelo de datos
Potencia Reactiva línea U	React_Pw_1 / 1000 (De VAr a kVAr)
Potencia Reactiva línea V	React_Pw_2 / 1000 (De VAr a kVAr)
Potencia Reactiva línea W	React_Pw_3 / 1000 (De VAr a kVAr)
Potencia Activa línea U	Act_Pw_1 / 1000 (De W a kW)
Potencia Activa línea V	Act_Pw_2 / 1000 (De W a kW)
Potencia Activa línea W	Act_Pw_3 / 1000 (De W a kW)
Voltaje línea U	V1
Voltaje línea V	V2
Voltaje línea W	V3
Corriente línea U	A1
Corriente línea V	A2
Corriente línea W	A3

El tratamiento de datos que se realiza para mostrar la información en el panel de información (objeto InfoPanel) del dispositivo “Grid” y en los gráficos de barras (objeto BarGraph) se muestra en la Tabla 6.

Tabla 6: Tratamiento de datos realizado para el objeto InfoPanel a través de los datos recibidos por el dispositivo Grid y para los gráficos de barras del objeto BarGraph

Magnitud	Conversión realizada desde el modelo de datos
Potencia del equipo línea U	Grid_PU_Set / 1000 (De W a kW)
Potencia del equipo línea V	Grid_PV_Set / 1000 (De W a kW)
Potencia del equipo línea W	Grid_PW_Set / 1000 (De W a kW)
Potencia Reactiva línea U	(PS_QU_Grid + React_Pw_1) / 1000 (Suma “Grid” y “Analyzer” y conversión de VAr a kVAr)
Potencia Reactiva línea V	(PS_QV_Grid + React_Pw_2) / 1000 (Suma “Grid” y “Analyzer” y conversión de VAr a kVAr)
Potencia Reactiva línea W	(PS_QW_Grid + React_Pw_3) / 1000 (Suma “Grid” y “Analyzer” y conversión de VAr a kVAr)
Potencia Activa línea U	(PS_PU_Grid + Act _Pw_1) / 1000 (Suma “Grid” y “Analyzer” y conversión de W a kW)
Potencia Activa línea V	(PS_PV_Grid + Act _Pw_2) / 1000 (Suma “Grid” y “Analyzer” y conversión de W a kW)
Potencia Activa línea W	(PS_PW_Grid + Act _Pw_3) / 1000 (Suma “Grid” y “Analyzer” y conversión de W a kW)
Voltaje línea U	PS_VU_Grid
Voltaje línea V	PS_VV_Grid
Voltaje línea W	PS_VW_Grid
Corriente línea U	PS_IU_Grid + (A1 * PS_PU_Grid / abs(PS_PU_Grid)) (Corriente de “Analyzer” multiplicado por el signo de la potencia activa de “Grid” y posterior suma de la corriente de “Grid”)
Corriente línea V	PS_IV_Grid + (A2 * PS_PV_Grid / abs(PS_PV_Grid)) (Corriente de “Analyzer” multiplicado por el signo de la potencia activa de “Grid” y posterior suma de la corriente de “Grid”)
Corriente línea W	PS_IW_Grid + (A3 * PS_PW_Grid / abs(PS_PW_Grid)) (Corriente de “Analyzer” multiplicado por el signo de la potencia activa de “Grid” y posterior suma de la corriente de “Grid”)

Anexo G. Pruebas de usabilidad y guion de prueba manual

En el apartado 4.4 se han expuesto las pruebas realizadas en la aplicación. En este anexo se habla de medidas a tener en cuenta respecto a la usabilidad con los objetos de realidad aumentada utilizando Vuforia. También se incluye el guion que se sigue para realizar una prueba manual en la aplicación.

Conclusiones de las pruebas de usabilidad sobre Vuforia

Un punto importante que afecta directamente al reconocimiento del entorno es el uso de la opción "*Extended Tracking"* [14] de Vuforia. Esta habilita a los objetos virtuales a estar disponibles sobre la pantalla del dispositivo incluso si la baliza o marca que se utiliza para reconocer el entorno se vuelve inaccesible para la cámara. Esto hace que los modelos 3D no estén sujetos solo a la baliza, si no que el reconocimiento del entorno real tiene que ser más exhaustivo para colocar los objetos sobre él y preparar la aplicación ante el posible movimiento de la cámara.

A partir de la documentación de Vuforia y con el feedback recibido de las demos que se fueron generando, se llegó a las siguientes conclusiones:

- La **baliza** [15] (también llamada marca o imagen de reconocimiento) debe ser rica en detalle (apareciendo diferentes objetos o formas en ella), debe tener un buen contraste de color entre claros y oscuros y no debe contenga patrones repetitivos. Vuforia analiza las imágenes que actúan como baliza añadiendo puntos de reconocimiento entre los puntos de contraste que encuentra sobre las imágenes. Además, puntúa las imágenes de 1 a 5, cuanta más alta sea la puntuación mayor estabilidad y rendimiento tendrá la aplicación.

- Al igual que para la baliza es importante generar un buen contraste, con el **entorno real** ocurre lo mismo. Aunque a priori puede parecer que un fondo liso es perfecto para utilizar la aplicación, no lo es. Vuforia realiza un reconocimiento del entorno a través de la cámara del dispositivo y toma referencias para situar los objetos 3D. Si el fondo con el que se enfoca con la cámara para situar la aplicación tiene irregularidades y es complejo, la imagen será más estable ya que contará con más puntos de referencia.

- Cambiar el dispositivo móvil de posición moviéndolo lateralmente desestabiliza los puntos de referencia tomados al hacer haber hecho previamente el reconocimiento de la baliza. El movimiento correcto es el de **rotación** manteniendo el dispositivo fijo en un punto.

- La **distancia con la baliza** también puede causar problemas en la usabilidad. Cuanto mayor sea esta distancia, más pequeños se dibujarán los objetos virtuales sobre la pantalla y viceversa. Lo apropiado para esta aplicación es que el dispositivo solo gire lateralmente y los espacios aparezcan enteros en la pantalla del dispositivo. Dependiendo del tamaño de la baliza, la distancia óptima varía. Por ejemplo, si el tamaño de la imagen de reconocimiento es DIN A4, la cámara del dispositivo debe situarse a aproximadamente 60cm.

Guion de prueba manual sobre la aplicación

A continuación, se incluye el guion con los pasos a seguir para realizar esta prueba:

1. Colocar la baliza sobre una mesa u otra superficie y apuntar con la aplicación a ella. En ese momento la aplicación iniciará el reconocimiento de la imagen y del entorno real que apunta la cámara.
2. Ajustar la distancia del dispositivo móvil con la baliza para observar los objetos 3D correctamente. La posición óptima al enderezar el dispositivo es cuando se ve el objeto que representa al equipo entero en la pantalla como en la Figura 39. Si el equipo está en estado “Funcionando”, se puede seguir la prueba manual.

Figura 39: Objeto 3D que representa el estado del equipo

3. Girar el dispositivo hacia la derecha hasta visualizar el panel de información al completo como en la Figura 40. Comprobar que hay valores en todos los campos de texto y esperar unos segundos hasta que haya una actualización de datos y comprobar que se actualizan automáticamente.

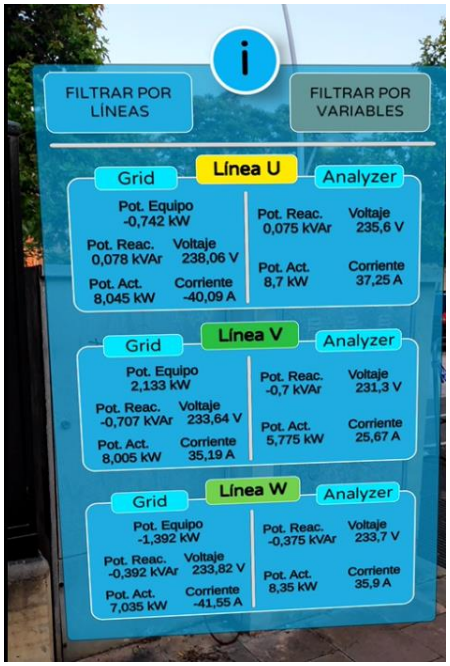


Figura 40: Panel de información

4. Pulsar en el botón “Filtrar por variables” para cambiar la vista del panel de información y comprobar que el resto de campos de texto también tienen valores.
5. Girar el dispositivo móvil hacia el equipo otra vez y pulsar en el botón de “Gráficos” en el menú superior.
6. Aparecerá un gráfico de barras sobre la pantalla como en la Figura 41.

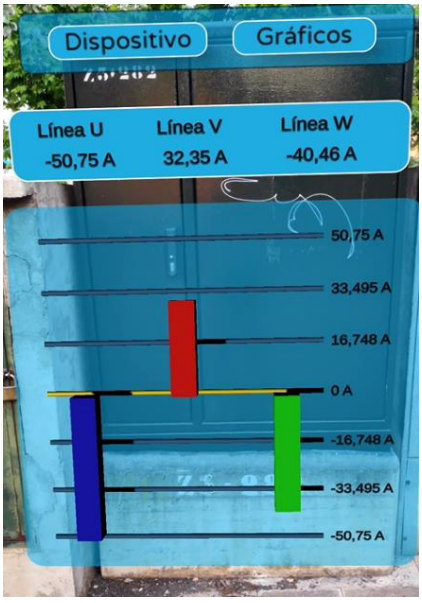


Figura 41: Gráfico de barras

7. Girar el dispositivo hacia la izquierda hasta que aparezca el panel de actuación de los gráficos como se muestra en la Figura 42. Pulsar sobre uno de los botones que indican otra magnitud y comprobar que el gráfico cambia. Esperar unos segundos hasta que haya una actualización de datos y comprobar que el gráfico cambia automáticamente. Comprobar también que en la parte derecha sigue apareciendo el panel de información.

Figura 42: Panel de actuación de los gráficos

8. Pulsar en el botón “Gráfico de líneas” que aparece en el panel de la Figura 42 para comprobar que el gráfico mostrado cambia como en la Figura 43.

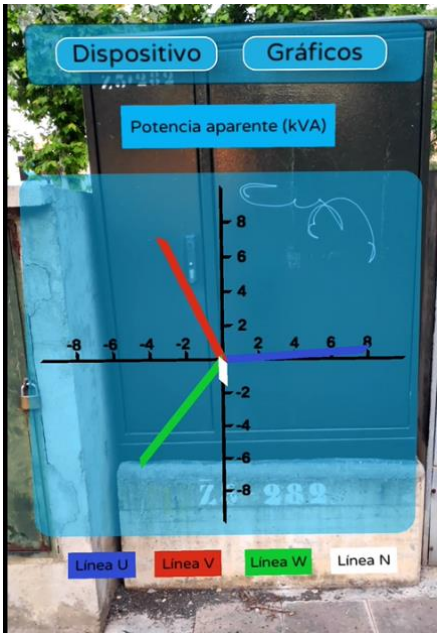


Figura 43: Gráfico de líneas

9. Abrir el menú rígido inferior como se muestra en la Figura 44, y pulsar sobre el botón “Dispositivo” para cambiar el modo de visualización y volver a la ventana inicial.

Figura 44: Menú rígido inferior

10. Girar el dispositivo móvil hacia la izquierda para visualizar el panel de actuación sobre el estado del equipo como en la Figura 45.

Figura 45: Panel de actuación sobre el estado del equipo

11. Interactuar con los botones del panel para visualizar como cambia el estado en la representación del equipo en el espacio central.

Anexo H. Manual de usuario

Contenido

Gonzalo Berné Melguizo

1. Instalación

La instalación de la aplicación se realiza a través de un fichero “.apk” en dispositivos móviles o tablets con versión Android superior a la v6.

Es necesaria la instalación de una aplicación para conectarse a una red VPN como “WireGuard” [16]. Una vez instalado, hay que configurar una nueva conexión.

2. Reconocimiento

La aplicación despliega elementos virtuales en el espacio visualizado por la cámara del dispositivo. Para ello, el dispositivo debe leer y reconocer una imagen o baliza como se ve en la Imagen 1.

Imagen 1: Reconocimiento de la aplicación a través de la imagen

Imagen de reconocimiento o baliza disponible en el apartado 8.

3. División de la interfaz de usuario

Los elementos virtuales de la aplicación se dividen en tres espacios, izquierdo, central y derecho:

- El espacio izquierdo sirve como panel de actuación para actuar sobre lo que se está visualizando en el espacio central (Imagen 3).
- El espacio central muestra el estado del dispositivo o gráficos (Imagen 2).
- El espacio derecho está formado por un panel de información donde se muestran las magnitudes más importantes del equipo que se está monitorizando (Imagen 4).

La aplicación cuenta con dos modos de visualización, “Monitorización del equipo” y “Gráficos”. A través de los menús se puede cambiar de un modo de visualización a otro depende de si el usuario quiere ver el estado del equipo o si desea ver los gráficos. El espacio izquierdo cambia según el modo de visualización escogido pero el espacio derecho siempre muestra el mismo panel de información.

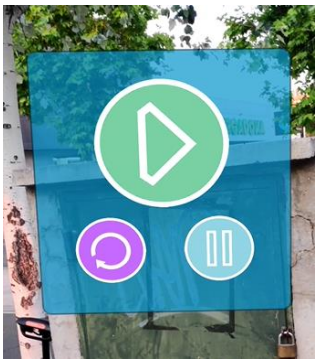


Imagen 3: Panel de actuación del espacio izquierdo

Imagen 2: Monitorización del equipo en el espacio central

Imagen 4: Panel de información en el espacio derecho

4. Monitorización del equipo

El modo de visualización “Monitorización del equipo” es el activado por defecto cuando se inicia la aplicación. En el espacio central se muestra el equipo en el estado en el que se encuentra y sus líneas (U, V, W y Neutro). El equipo puede estar en cuatro estados diferentes:

Funcionando En este estado, el equipo se muestra con icono con un “tick” verde. Además, indica de que líneas está consumiendo potencia (color amarillo) y en cuales está inyectando (color verde).	

Pausa Cuando el equipo está detenido, se muestra con un icono en pausa.	

Reiniciando Si el equipo está reiniciándose para volver a ponerse en funcionamiento, el icono que muestra el equipo es una flecha que gira sobre si misma hasta que el equipo cambia de estado.	

Error Si el equipo se encuentra en estado de error, este se muestra con un icono sobre fondo rojo con una X.	

El panel de actuación que aparece en el lado izquierdo con el modo de visualización "Monitorización del equipo" cuenta con 3 botones para cambiar de estado el equipo. Además, como se muestra en la Imagen 5, si el equipo está en estado de error, muestra un panel con la información del error.

Imagen 5: Panel de actuación sobre el estado del equipo y mensaje de error

5. Gráficos

Cuando a través de los menús se cambia al modo de visualización de gráficos, en el espacio central se pueden visualizar dos tipos de gráficos: de barras (Imagen 6) o de líneas (Imagen 7).

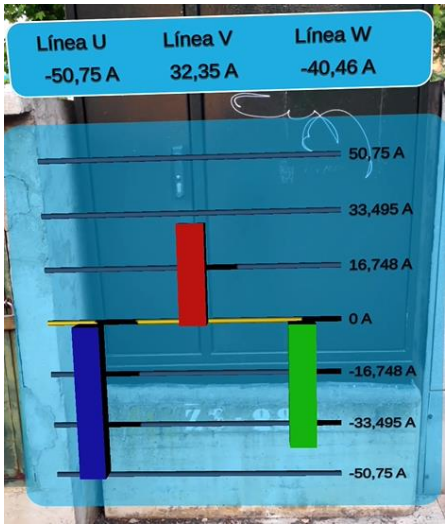


Imagen 6: Gráfico de barras

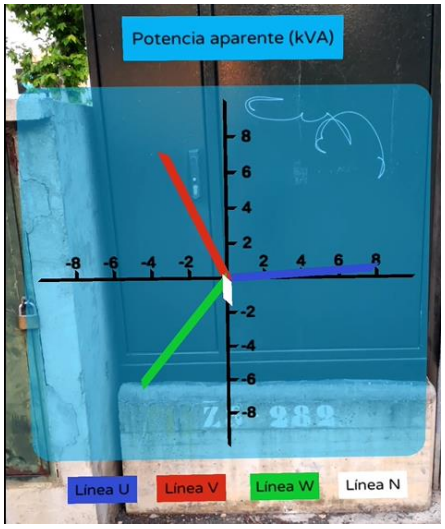


Imagen 7: Gráfico de líneas

A través del panel de actuación del espacio izquierdo, se puede elegir la magnitud a visualizar en el gráfico de barras entre (Imagen 9):

Potencia de equipo (kW)	Potencia reactiva (kVAr)	Potencia activa (kW)
Voltaje (V)	Corriente (A)	

El gráfico de líneas muestra la Potencia Aparente del equipo. A través del panel de actuación se puede elegir ver (Imagen 8):

- La Potencia Aparente (kVA) de las líneas antes de ser balanceada por el equipo.
- La Potencia Aparente (kVA) de las líneas después de ser balanceada por el equipo.

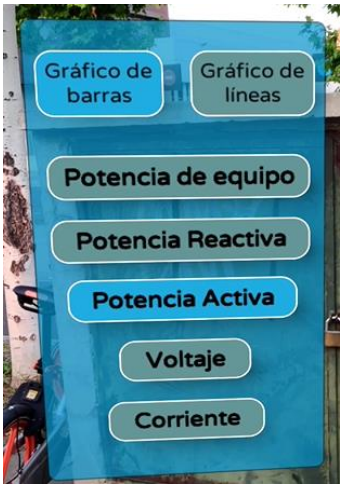


Imagen 9: Panel de actuación gráfico de barras

Imagen 8: Panel de actuación gráfico de líneas

6. Panel de información

El panel de información siempre ocupa el espacio derecho independientemente del modo de visualización seleccionado. Este panel muestra los datos medidos por los dispositivos Analizer (antes de pasar por el equipo) y Grid (después de pasar por el equipo) del equipo. El panel permite mostrar una visión global de todas las variables al seleccionar el botón “Filtrar por líneas” (Imagen 11) o solo mostrar una variable en concreto con el botón “Filtrar por variable” (Imagen 10). En la primera opción, el color del nombre de cada línea será amarillo (si el equipo está consumiendo potencia de ella) o verde (si el equipo está inyectando).
Las diferentes magnitudes que muestra el panel son:

Potencia de equipo (kW)	Potencia reactiva (kVAr)	Potencia activa (kW)
Voltaje (V)	Corriente (A)	

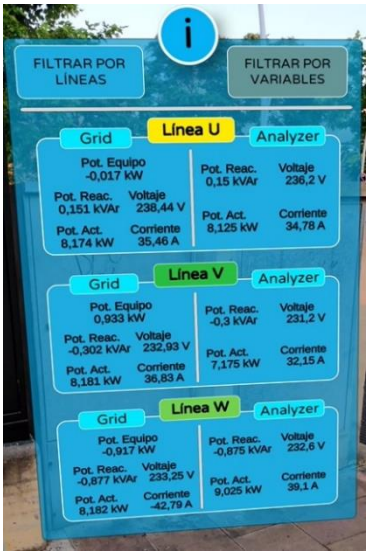


Imagen 11: Panel de información filtrando por líneas

Imagen 10: Panel de información filtrando por variables

7. Menús

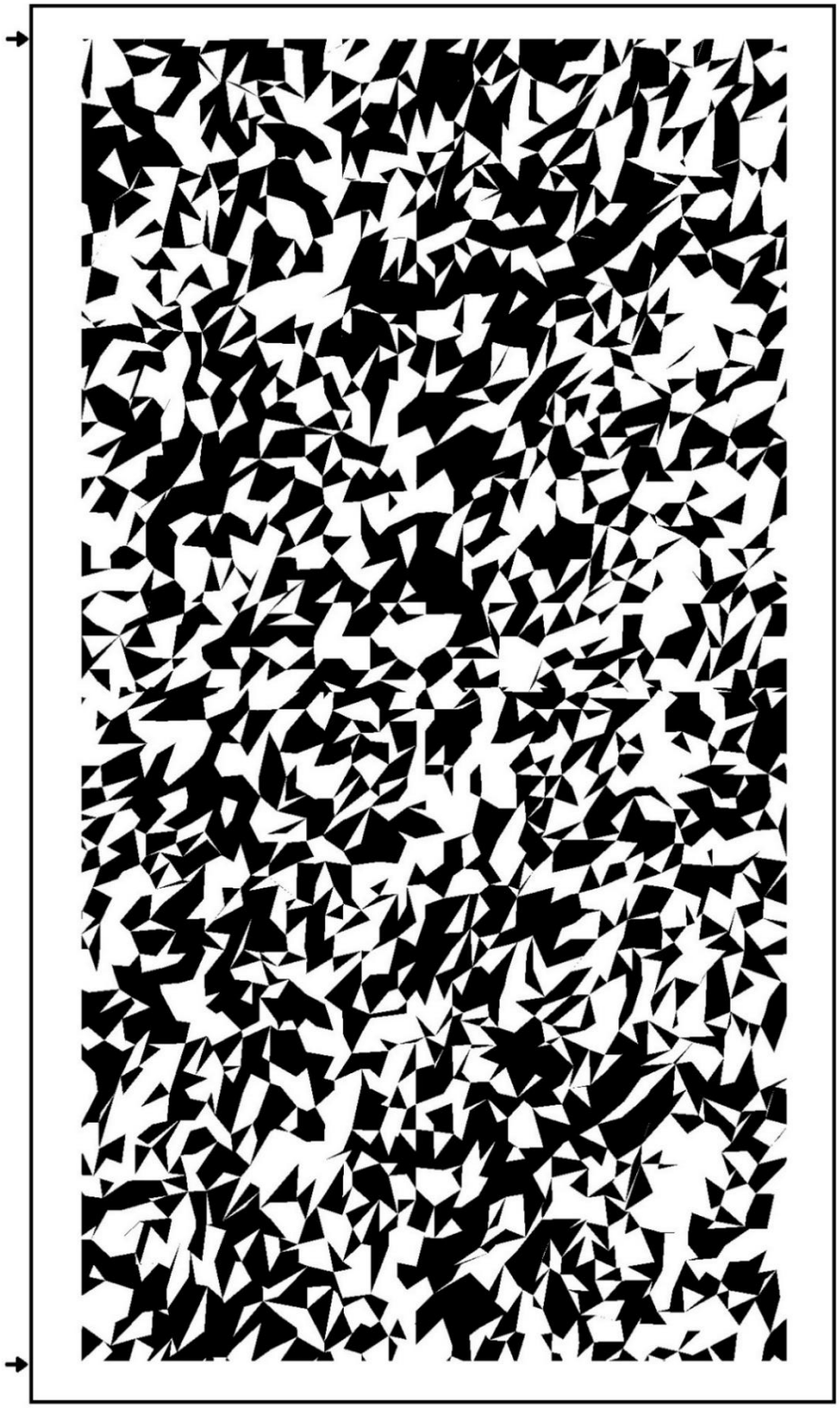
La aplicación cuenta con dos menús de control que cambian entre un modo de visualización y otro (Monitorización del equipo y Gráficos).

- Un menú superior situado encima del equipo o de los gráficos en el espacio central (Imagen 12).
- Un menú rígido (que está siempre sobre la pantalla, no depende del entorno) situado en la parte inferior de la pantalla (Imagen 13).

Imagen 12: Menú superior

Imagen 13: Menú rígido cerrado y abierto

8. Baliza o imagen de reconocimiento



Referencias (Anexos)

[9] modbus.org (2002). MODBUS over Serial Line Specification & Implementation guide V1.0. https://modbus.org/docs/Modbus_over_serial_line_V1.pdf

[10] Unity corrutinas. https://docs.unity3d.com/Manual/Coroutines.html [Ultimo acceso Junio 2021]

[11] 3D Bar Chart. https://assetstore.unity.com/packages/tools/gui/3d-interactive-barchart-157887 [Ultimo acceso Junio 2021]

[12] Lean GUI. https://assetstore.unity.com/packages/tools/gui/lean-gui-72138 [Ultimo acceso Junio 2021]

[13] Clean Vector Icons. https://assetstore.unity.com/packages/2d/gui/icons/clean-vector-icons-132084 [Ultimo acceso Junio 2021]

[14] Vuforia Extended Tracking. https://library.vuforia.com/features/environments/device-tracker-overview.html [Ultimo acceso Junio 2021]

[15] Buenas prácticas para reconocimiento de imágenes. https://library.vuforia.com/features/images/image-targets/best-practices-for-designing-and-developing-image-based-targets.html [Ultimo acceso Junio 2021]

[16] WireGuard: fast, modern, secure VPN tunnel. https://www.wireguard.com/ [Ultimo acceso Junio 2021]